Guidance and Control of Sounding Rockets

David Andrew Wright

A dissertation submitted to the Department of Electrical Engineering,
University of Cape Town, in fulfilment of the requirements
for the degree of Master of Science in Engineering.

Supervisors:
Dr Peter Martinez
Professor Michael Inggs

Cape Town, February 2013

Declaration

I know the meaning of plagiarism and declare that all the work in the document, save for that which is properly acknowledged, is my own.

Signature of Author . . . . . .

Cape Town
11 February 2013

Abstract

This dissertation presents the design, fabrication and testing of a sounding rocket flight computer for the South African Astronomical Observatory (SAAO).

Sounding rockets carry instruments with which to take measurements in the Earth's atmosphere in sub-orbital flight. The South African Astronomical Observatory (SAAO) requires a flight computer for their sounding rockets. This flight computer is to replace the current commercial flight computer currently in use improving on its functionality and expandability.

Based on SAAO's requirements a specification has been derived. This specification includes the functions required of the flight computer, the stresses and constraints the flight computer must operate under and the testing procedures to verify the operation of the flight computer. From this specification a modular and extensible flight computer has been designed and implemented. The design included the design of the hardware as well as the design of the operating system and algorithms to run on the flight computer hardware. In order to meet the requirements of SAAO a master-slave architecture was chosen in the design using the Controller Area Network bus.

Once designed the flight computer was prototyped and tested using tests outlined in the specification. The hardware has been taken through three prototyping phases, with each phase bringing the flight computer closer to meeting the specification. Algorithms were prototyped and simulated on a PC in the C language before being ported to embedded C with fixed point arithmetic to run on the hardware. Simulations were carried out with actual flight data obtained from the RDAS mailing list. Also developed was a graphical user interface, using the Qt framework in the C++ language, to

interface with the flight computer and to display simulations results.

Acknowledgements

Soli Deo Gloria

I would like to acknowledge thankfully the contributions of a number of people without whom this dissertation would not have been possible.

Firstly, I would like to acknowledge the contribution of Dr Peter Martinez in the creation of this dissertation. Dr Martinez has patiently guided the content of this dissertation through many meetings and with invaluable feedback on my design choices. He has facilitated many great opportunities to flight test my work at OTB, and to present my work. I am especially grateful to have been given the opportunity to pursue a masters in space related engineering.

Professor Michael Inggs of the RRSG group has enabled me to pursue this masters, by agreeing to supervise me and arranging funding. Professor Inggs has guided the structure and content of this dissertation in feedback on many drafts.

Tellumat for their provision of funding for the project as well as providing free manufacturing services. Dave Jackson, Shareef Hoosain and Ismail Fakier at Tellumat who arranged for manufacturing, and provided much needed advice on the hardware.

Bruce and Jeff Cairns who have provided invaluable technical help in materials and machining. Invariably when I'm stuck with a project they are happy to help and provide much needed advice. Jeff has stayed up in the middle of the night to help me meet deadlines a number of times. Nicci Cairns who has kept me very well fed and been my second mom.

Craig Tong at RRSG has been an invaluable source of C and C++ knowledge. Many a time have I been boggled by a statement in C++ and Craig has come to the rescue!

My Mom, Dad and Brother for providing me the opportunity to study as well as giving me their love and support. They have unfailingly supported me through my academic carrier and I am exceptionally thankful to the Lord for his gift in this.

Finally, my wife-to-be-Beth Cairns for her love, her belief in me and her support through the most challenging moments. For her, I am also exceptionally thankful to the Lord.

Contents

List of Figures

List of Tables

Nomenclature

Apogee — The highest altitude in a rocket's trajectory

CAN — Controller Area Network bus

COTS — Commercial off-the-shelf components

dB — Decibels

EEPROM — Electrically Erasable Programmable Read-Only Memory

FR4 — A fibreglass dielectric substrate used between copper layers of a printed circuit board

FTDI — Future Technology Devices International - A manufacturer that produces USB to serial converters

GPS — Global Positioning Satellite system

GUI — Graphical User Interface

IC— Integrated Circuit

IMU — Inertial measurement unit

ISM — Industrial, scientific and medical radio bands

ITAR — International Traffic in Arms Regulations

JTAG — Joint Test Action Group - A method of flashing and debugging micro-controllers

LCD — Liquid Crystal Display

LNA — Low Noise Amplifier

MEMs — Micro-electromechanical systems

MIPS — A unit of computing speed equivalent to a million instructions per second

Mutex — Mutual Exclusion Object

NMEA — National Marine Electronics Association; when used to refer to GPS data, it refers to the format of the data stream output by the GPS receiver

PA — Power Amplifier

POSIX — Portable Operating System Interface

RAM — Random Access Memory

RF — Radio Frequency

ROHS — Restriction of hazardous substances directive

SPDT — Single pole double throw switch

TXRX — Transceiver

µC — Micro-controller

USB — Universal Serial Bus

Chapter 1

Introduction

This dissertation presents the design, fabrication and testing of a sounding rocket flight computer for the South African Astronomical Observatory (SAAO).

This chapter will begin with an introduction to sounding rocketry in order to familiarise the reader with the application of the flight computer. This will be followed by a discussion of the background to the project, an explanation of the objective of the project and finally an overview of this dissertation.

1.1 Sounding Rockets

Sounding rockets carry instruments with which to take measurements in the Earth's atmosphere in sub-orbital flight. The name is derived from the nautical term "to sound", meaning to measure [1]. The typical trajectory of a sounding rocket as shown in figure 1.1 is parabolic. This means that the initial trajectory of the rocket is near vertical, ending in what appears to be a brief pause at apogee before descending back to the Earth's surface along a trajectory dependent on the recovery systems and atmospheric conditions. This highly vertical sub-orbital trajectory means that it is possible to recover the rocket near the launch site. Further, because the rocket is only required to be sub-orbital it is possible to use small rocket motors. This makes sounding rockets

the ideal platform for low-cost, high-altitude experiments [2].

For a rocket to function as a sounding rocket it requires the ability to detect in-flight events such as apogee and to initiate in-flight events such as the deployment of parachutes. Without this capability it would not be possible to recover the rocket or the measurement data from any on-board instrumentation. In order to provide this capability a rocket requires a flight computer. This may be as simple as an altimeter measuring only atmospheric pressure in order to detect apogee and control the deployment of parachutes, or as complex as a six-degree-of-freedom inertial measurement unit with support for external experimental payloads, cameras, GPS, telemetry, multiple igniter firing channels and in-flight user control.

In summary, a sounding rocket flight computer is required to facilitate two major functions. Firstly it must facilitate measurement of flight performance and various atmospheric variables and secondly it must respond to this measured data to detect and initiate in-flight events such as launch, stage separation, stage ignition and the launch of recovery systems.

1.2 Project Background

Each year the South African Astronomical Observatory (SAAO) runs a space technology course in conjunction with the University of Cape Town. As part of this course the students design an electronics payload for a sounding rocket. Up until now the project has used a commercially available flight computer to support their payloads, but this flight computer no longer meets the requirements of the rocket project. This is due to the fact that the flight computer is limited in its ability to measure inertial and atmospheric variables, is inflexible in terms of its ability to support external payloads and is unable to guarantee data integrity.

In a general survey of rocket flight computers it appears that manufacturers for the most part cater either for the amateur rocketry community or for the professional aerospace and defence community. This means that commercially available flight computers within the budgetary reach of small rocket projects are generally simplistic

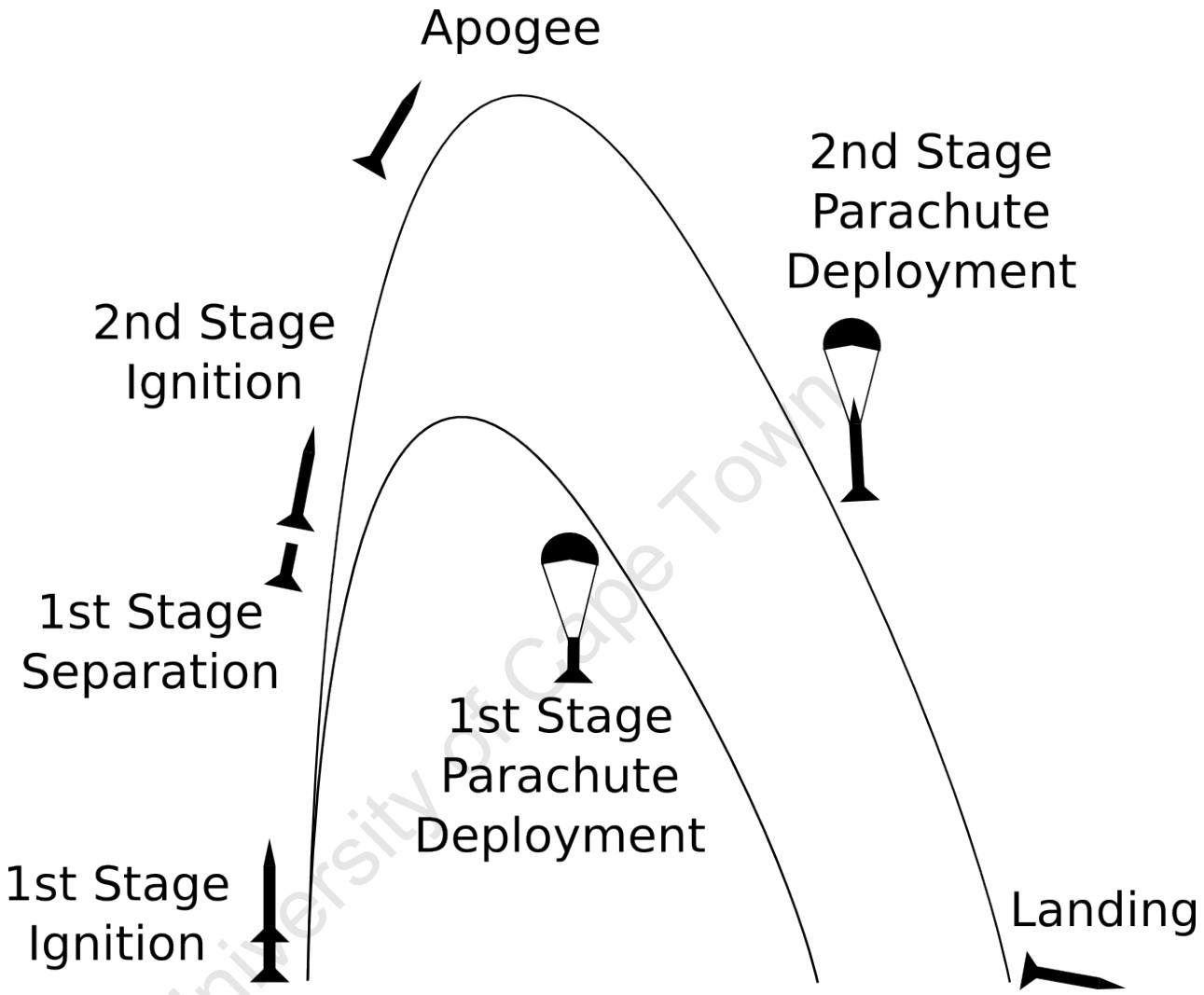

Figure 1.1: **Launch:** Launch commences with the ignition of the first stage motor. This involves the firing of pyrotechnics in order to start a rocket motor. **Separation:** If the rocket is a multi-stage stage rocket, it will separate from the burnt out first-stage rocket motor, generally using drag or a pyrotechnic charge. The burnt out first-stage will deploy a parachute to descend to the ground. **Stage Ignition:** Once separated the rocket will fire a pyrotechnic charge in order to ignite a new motor stage. The process of separation and stage ignition will be repeated until all the rocket motor stages have burnt out. **Apogee:** The rocket will continue up to its apogee, where it will deploy its parachutes. Normally a smaller drogue chute will be deployed first, followed by a much larger master parachute. **Landing:** The flight ends with the rocket landing on the ground and starting up recovery beacons to enable the recovery team to locate the rocket.

in nature, aiming to support the bare minimum of functions to enable rocket launch and recovery. Two such products are the GWIZ series of flight computers [3] and the very popular RDAS flight computer [4], which we shall review later. Societies or universities that desire more complex electronics to control their rockets generally design and build their own systems, as in the case of the Portland State Aerospace Society [5]. This is illustrative of SAAO's need for the design of their own flight computer. There simply is no commercially available flight computer that can fulfil their complex requirements, and instead of spending large sums of money on defence or space-grade flight computers they have chosen to build their own flight computer catering to their specific needs.

1.3 Project Objective

The objective of this dissertation is to produce a sounding rocket flight computer that meets the requirements of the SAAO. This includes a review and analysis of available flight computers, followed by the design, fabrication and testing of a flight computer to meet the requirements of the SAAO.

1.4 Dissertation Overview

Review of Similar Work

Once the documentation was in place describing what the flight computer was required to do, a review of flight computers, either available commercially or in development by other research institutions, was started. Two flight computers were selected for review, the RDAS flight computer [6], manufactured by AED electronics, and the Portland State Aerospace LV2b [7] flight computer. A summary of this review is given below; the full review may be found in chapter 2.

The RDAS was the commercially available flight computer that the SAAO had previously used for its rockets. It was found to have a number of shortcomings, such

as:

- A monolithic architecture in a failure-prone environment.
- An extensible architecture, where the extension does not improve redundancy.

Through the review of the RDAS a number of important lessons were learnt; these included:

- The importance of redundancy in a failure-prone environment.
- The importance of reconfigurability and extensibility in a reusable flight computer.
- The need for modularity to facilitate an extensible architecture.
- The importance of clear flight computer status indications in dangerous environments, such as a launch site.

The PSAS LV2b flight computer was developed by the Portland State Aerospace Society. It is similar to this project in that it is a university-built flight computer for an experimental sounding rocket program.

The review of the LV2b flight computer revealed a particular design choice this writer deems to be poor. The LV2b uses a master-slave architecture in which the main communication bus on the flight computer is USB. This introduces a single-point failure for the whole flight computer because the communication bus requires the USB hub to be functioning for any communication to take place.

A number of positive lessons were learnt from review of the LV2b; these included:

- The ability of the master-slave architecture to achieve extensibility and modularity.
- The review of communication buses, such as USB, informed selection of possible communication buses for the SAAO flight computer.

- The use of backup power supplies for critical systems to avoid mission failure in the event of a battery pack failure.
- The self-calibration of the inertial measurement unit via the temperature sensor was noted for possible inclusion in the SAAO flight computer.

Requirements, Functional Analysis and Specification

The SAAO provided a list of 21 requirements for their flight computer. These requirements were analysed for their functional content, and then broken down into the functions that are required of the flight computer. Table 1.1 shows the breakdown of these functions with reference to the requirements from which they are derived. A full discussion of the SAAO requirements and their functional breakdown may be found in chapter 3.

Design

Having reviewed some existing flight computers, the design of SAAO flight computer began in earnest. The design and fabrication philosophy used in this project was the spiral model. This entailed a cycle of design, fabrication, testing and redesign [8]. The following is a summary of the design of the SAAO flight computer. A full description of the design may be found in chapter 4.

Before each of the individual functions could be designed for, the overall system architecture needed to be decided upon. To this end, four major architectures were reviewed and the master-slave architecture was chosen as the best suited architecture to meet the specification. Further, the Controller Area Network communication bus (CAN bus) was chosen from amongst a number of alternative bus architectures including I2C and RS - 485. Figure 1.2 is a diagram showing the final system architecture.

The master-slave architecture and the requirement for modularity necessitate that each function or associated group of functions be assigned to an individual node. Each node in the flight computer has a repeated hardware base including a processor and a CAN

Function Reference	**Function Description**	**Requirement Reference**
F1	Measurement of acceleration in X, Y, Z, Pitch, Yaw and Roll	R1
F2	Measurement of pressure	R1
F3	Determination of position via GPS, F1 and F2	R1
F4	Measurement of temperature	R1
F5	Measurement of flight computer currents and voltages	R5, R7
F6	System status monitoring	R5, R7, R15
F7	F1 - F6 Stored in persistent memory	R8, R16
F8	F1 - F6 Transmitted via wireless link	R3, R10
F9	Detection of launch	R2.1
F10	Detection of motor burnout	R2.2
F11	Detection of separation	R2.3
F12	Detection of new stage ignition	R2.4
F13	Detection of apogee	R2.5
F14	Detection of landing	R2.6
F15	Detection of trajectory outside of defined trajectory	R11
F16	Launch initiation	R9
F17	Separation of rocket stages	R2.2, R9
F18	Ignition of rocket stage motors	R2.3, R9
F19	Ground station display of real-time flight data	R6, R10
F20	Flight computer control via the ground station	R6, R10
F21	Audible indication of status	R5

Table 1.1: Summary of the functional requirements of the flight computer.

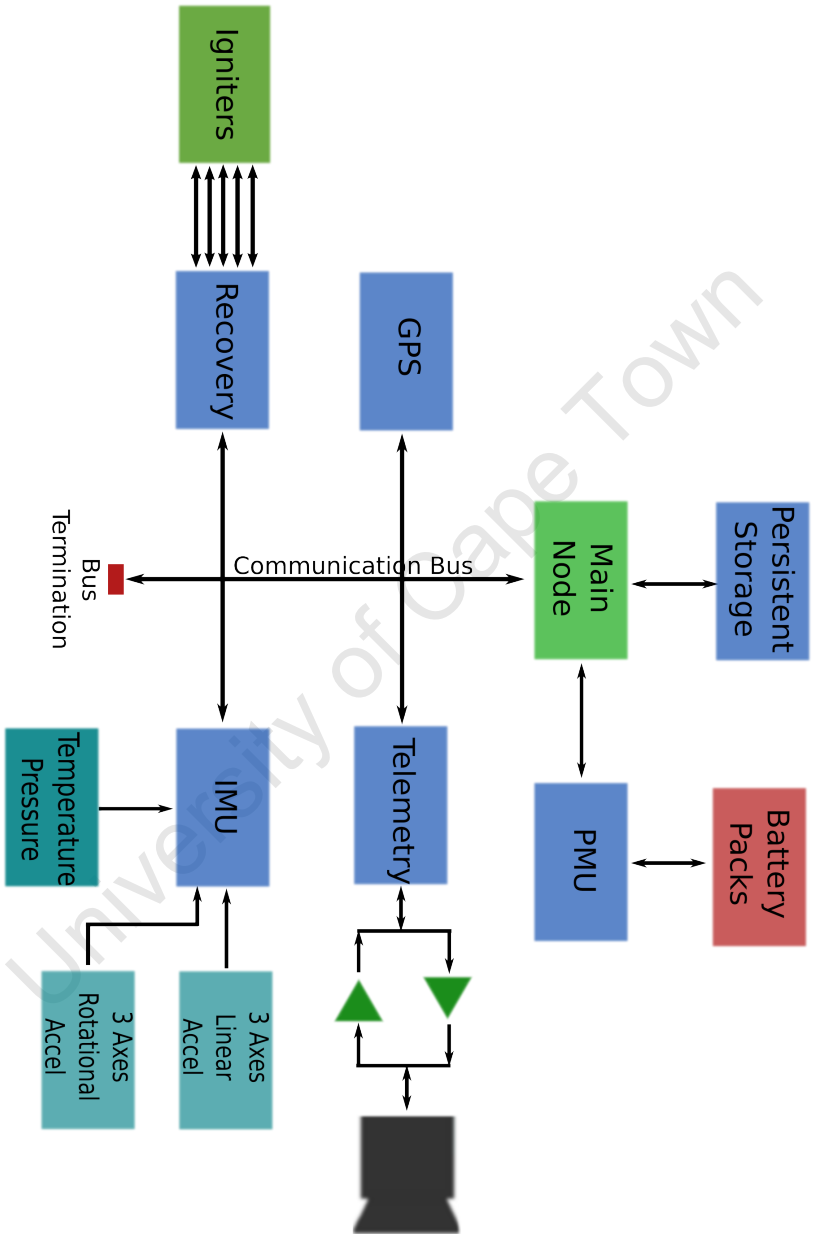

Figure 1.2: Master-slave architecture for the SAAO flight computer.

transceiver to provide access to the bus.

The flight computer consists of five nodes:

- The Master Node - This node deals with power distribution and data storage.
- The Recovery Node - This node deals with the firing of igniters to respond to in-flight events.
- The GPS Node - This node receives and processes GPS information.
- The Telemetry Node - This node sends and receives data from a ground station via a wireless communication link.
- The Guidance and Control Node - This node measures inertial information in 6 degrees of freedom, processes it and provides the user with the ability to control actuators with which to affect the flight path.

A sixth independent node is designed to serve as a ground station and includes GPS measurement and telemetry.

The firmware for the flight computer uses a time-triggered architecture [9]. This architecture was chosen over a pre-emptive architecture because it provides increased reliability and decreased system overhead. The flight computer uses a shared clock scheduler in order to synchronise the clocks of each of the nodes [9][10]. This provides a way to control communication on the CAN bus in a time-triggered manner.

RF Design of the Telemetry Output Stage

The purpose of the telemetry output stage is to increase the range of the telemetry. The output stage was designed and simulated using Agilent's Gensys. The RF output stage uses a switching architecture to incorporate both a power amplifier and a low noise amplifier onto the output and input of the RF transceiver. The switching is controlled by the RF transceiver; when the transceiver is in transmission mode it switches the output stage to the power amplifier channel and when it is in receive mode it switches

Amplifier	S11 (dB)	S21 (dB)	Isolation (dB)
PA	-25.763	21.189	
LNA	-22.248	22.409	
SPDT	-19.7	-0.35	-26

Table 1.2: Simulated RF output stage S parameters.

the output stage to the low noise amplifier channel. Table 1.2 shows the expected S parameters of the output stage.

Fabrication, Firmware/Software Development and Testing - Where good ideas face reality

Following the design of the SAAO flight computer, a number of prototypes were fabricated. Each of these prototypes was then tested for workmanship errors and fundamental design flaws. A brief overview of the fabrication, firmware development, testing and the results obtained is given below, a full description may be found in chapter 5.

Initial testing was carried out by writing test firmware for each of the peripherals on the node micro-controllers. In order to test the power distribution and measurement functions the master node was linked to resistive loads to verify the ability to both switch the power channels on and off and to measure the current consumption on each channel.

Although vibration and temperature testing are essential in the development of a flight computer for a rocket application, the lack of access to facilities in the time frame of this masters prevented this testing. The only environment-based testing done was vacuum testing. A board consisting of the node micro-controller and supporting electronics was placed in a vacuum chamber in order verify that the electronics would not out-gas in a vacuum. During the test the electronics continued to function correctly under the vacuum.

A graphical user interface (GUI) was written to test the functionality of the flight computer and to provide a platform to run simulations both on a PC and in the loop with the flight computer. The GUI was written using the QT framework in the C++ language. The graphical user interface provides manual control of the flight computer power management system and the recovery node igniter control system. It also shows the current inertial and power status of the flight computer. This information is displayed in LCD displays as voltages, currents, acceleration, velocity and position, and as real-time plots of acceleration and altitude.

In order to develop working algorithms for Kalman filtering and event detection, simulation code was written on a PC. Simulations were written in the C language and run using real recorded rocket flight datasets obtained from the RDAS flight computer mailing list. Event detection times were compared to the times given by the RDAS graphical user interface for the given dataset. Figure 1.3 shows the filtered and unfiltered data for the altitude of a rocket. The vertical green lines show the events with the time of detection and the event name.

The filtering and event detection algorithms were implemented in the inertial measurement unit by converting the floating-point arithmetic based algorithm into a fixed-point arithmetic based algorithm. The algorithms were then benchmarked using the same datasets used for the simulation. This was done to ensure that the algorithms were able to run at the required rate on the node micro-controller. The outputs of the implemented algorithms were also tested to ensure that they were the same as the simulated algorithms.

RF Implementation and Testing

The simulated RF designs were implemented and tested using a network analyser. Table 1.3 shows the S parameters obtained from the tests.

The results were not as good as the simulations. This may be explained by a number of factors. Among these is the inaccurate milling of the circuit boards and the assembly of the circuit boards by hand. In the case of the LNA, there was difficulty in tuning the input of the LNA as the electrical length of the input line was required to be very

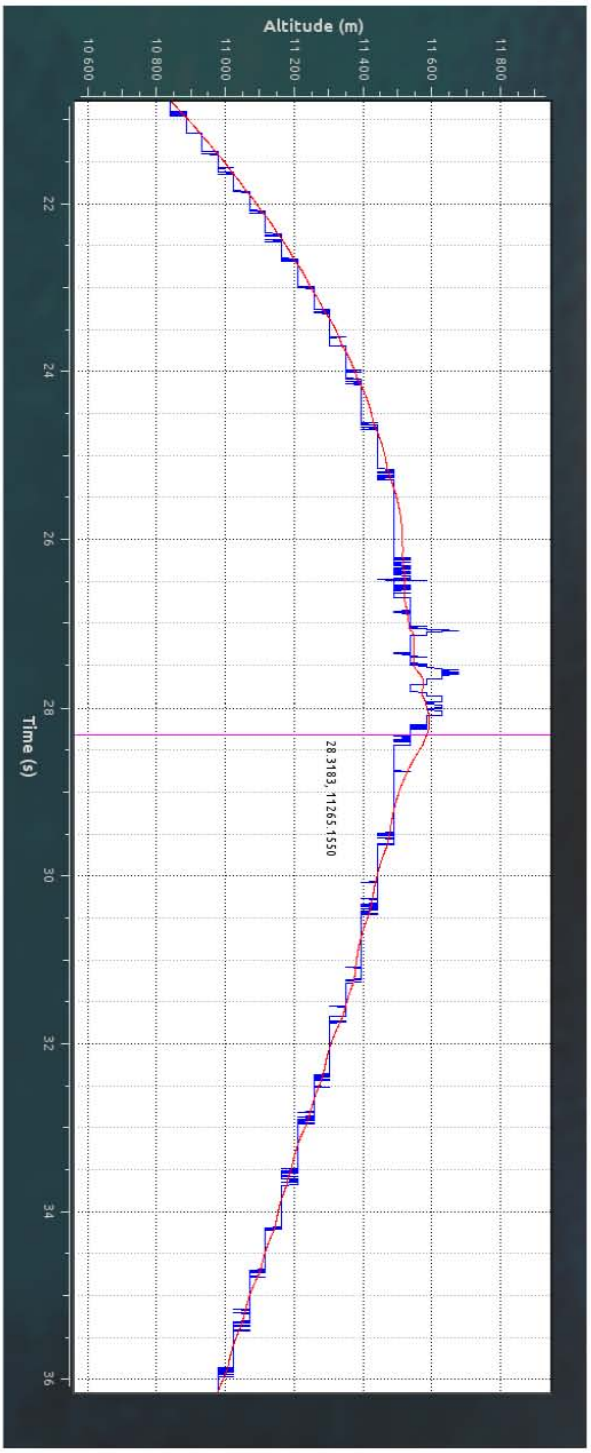

Figure 1.3: Plot of measured altitude (blue) and filtered altitude (red) of the apogee phase of a rocket with apogee detection shown at 28.318s by the magenta vertical line.

Amplifier	S11 (dB)	S21 (dB)	Isolation (dB)
PA	-16.412	18.988	
LNA	-14.610	27.509	
SPDT	-16.268	-0.7101	-21.074

Table 1.3: Measured RF ouptut stage parameters.

short. Finally, the S21 parameter of the SPDT is double the expected value because in the test set-up used the signal travelled through two SPDT switches.

Conclusions and Future Work

We have developed a rocket flight computer based on the requirements given by the SAAO. Each of the functions outlined in Table 1.1 was met, except for Functions 11, 12, 15, 17 and 18. These functions were not met either due to time constraints, or because they were dependent on functions omitted due to time constraints, for example Function 15, or due to a lack of multi-stage rocket flight data, as was the case with Functions 11, 12, 17 and 18. Not all the required testing was possible due to lack of access to facilities, such as a vibration chamber. Future work includes:

- Implementation of an attitude filter to fulfil Function requirement 15.
- Implementation of an algorithm to detect outside nominal trajectories as per Function 15.
- Implementation of algorithms to detect stage-separation and new stage-ignition as per Functions 11 and 12.
- Implementation of an algorithm for triggering separation and new stage-ignition as per Functions 17 and 18.
- Further testing and refining of the SAAO flight computer including simulation testing under rocket flight conditions.
- The refinement and extension of the graphical user interface to include three-dimensional trajectory plotting.

- Implementation of guidance and control of the rocket through the inertial measurement unit.

1.5 Conclusion

In summary of this chapter we have introduced the background and the objective of the project to reader. This was followed by a review of the relevant rocketry information regarding rocket flight profiles and in-flight events. We also gave the reader a brief overview of what to expect in this dissertation. In the following chapter we shall review some projects of similar intent.

Chapter 2

System Review of Currently Available Systems

2.1 Introduction

In this chapter we shall review two flight computers, giving an overview of their architectures and features. We shall also note the pros and cons of the design of each flight computer. Finally, the lessons learnt from each flight computer will be noted and carried forward into the design of the SAAO flight computer.

2.2 RDAS

Figure 2.1 shows the RDAS flight computer main circuit board. The RDAS flight computer is a commercially available flight computer aimed at the amateur rocketry community. It is manufactured in the Netherlands by a company called AED Electronics[6]. The review of this flight computer is especially applicable here because it is the flight computer that this dissertation aims to improve upon and replace.

The RDAS provides in-flight measurement of acceleration and pressure, which it uses to determine the altitude and apogee of the rocket. Based on the detection of apogee it

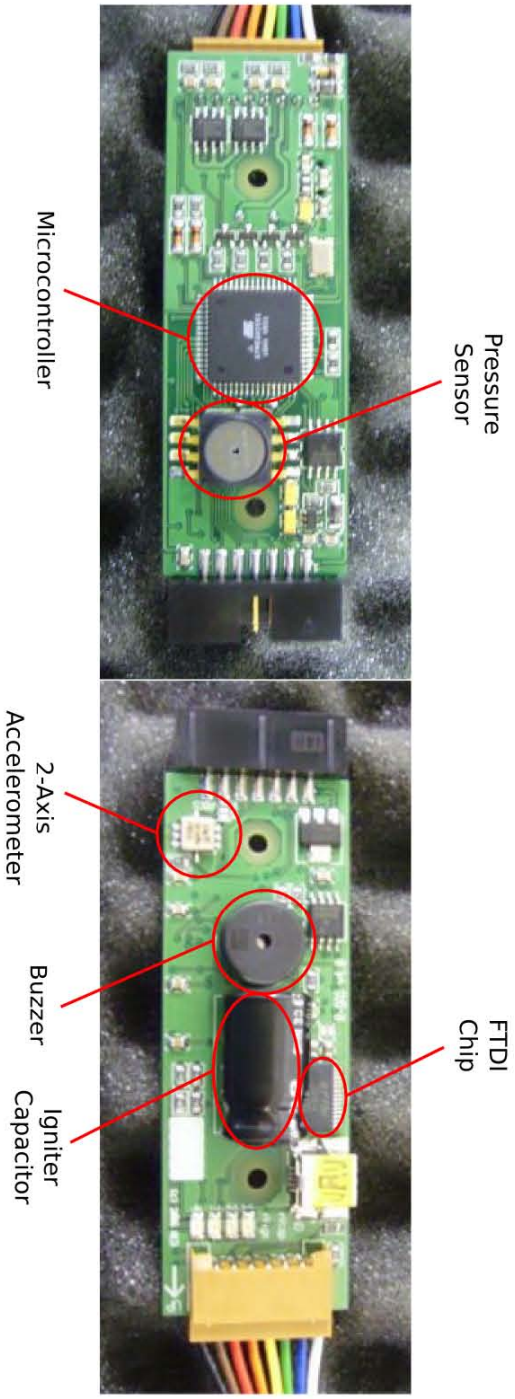

Figure 2.1: RDAS flight computer.

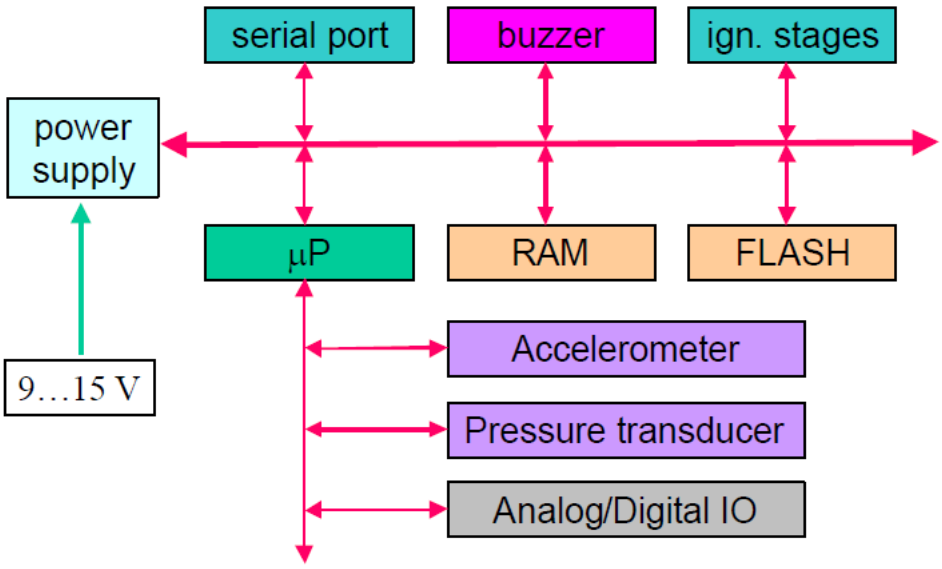

Figure 2.2: RDAS block diagram [6].

is able to deploy recovery systems through two igniter channels. These igniter channels can be configured to fire at the detection of apogee or after a user-defined time interval. There are also a number of user defined digital and analogue inputs to the RDAS allowing the user some flexibility in terms of the payloads the RDAS can support. Finally, through additional expansion nodes the RDAS is also able to provide GPS measurements and telemetry.

RDAS Architecture

The primary features of the RDAS flight computer are arranged in a monolithic architecture on a single circuit board. The advantage of this is that the flight computer occupies the least possible space. Figure 2.2 shows a block diagram of the RDAS flight computer.

The microprocessor controls all the operational functions in the above diagram, including the igniter stages, the buzzer, the memory, the inertial and pressure transducers and the additional analogue and digital I/O ports. Communication with the processor is facilitated through a serial connection and power is supplied through an independent

power supply.

Power Supply The RDAS requires a minimum voltage of 9V on its power supply which it then regulates down to the required DC voltages. The RDAS may also be powered through its USB port that is used for serial communication with the device.

Status Indication The RDAS uses a buzzer to convey a series of status-related messages to the user in a clearly audible manner. This is especially useful on the launch pad, when it is not possible to connect a computer to the RDAS to monitor its operation. It also provides an easy way of identifying fault conditions on the RDAS.

Inertial and Pressure Measurement Acceleration is measured in the Z axis, that is the axis parallel to the long axis of the rocket, on the main RDAS board with a range of $\pm$50g. The sensor is an analogue sensor with low power and low noise characteristics. These sensor features are important given the power constraints imposed by batteries and the need to determine the altitude of the rocket accurately and reliably when triggering recovery events. Pressure is measured with an absolute pressure transducer that is temperature-compensated over a range of -40$^{\circ}C$ to 125$^{\circ}C$. The sensor is however limited to measuring pressures down to a minimum of 20kPa. This limits the maximum altitude at which the sensor will function to a maximum of approximately 10km. Temperature compensation is important here as it allows for a more accurate measurement of altitude through the measurement of atmospheric ambient pressure. The pressure transducer provides a second measurement of altitude for redundancy. This illustrates the principle of redundant design.

Analogue and Digital Inputs The RDAS provides 4 digital TTL inputs to the user to support mission-related experiments. The RDAS logs the data that arrives on the port to EEPROM. The RDAS also provides 6 analogue inputs that are also logged to EEPROM. In this way the RDAS provides support for external payloads. It should be noted however that the limitation of this approach is that the data measured by these ports cannot be used to trigger any in-flight events through the RDAS. Therefore the

user cannot add further redundancy to the RDAS.

Memory The RDAS has enough persistent memory to store eight minutes of data from all of its input ports when they are sampled at a rate of 200Hz. This is the maximum available sampling frequency of the RDAS and once the sampling rate is set for a mission, it is fixed for the duration of the flight. This eight minutes puts an upper limit on the length of flight for which the RDAS may be deployed.

Extensible Architecture The RDAS is extensible allowing for the connection of telemetry, GPS and inertial measurements in additional axes. These additions are done via the I2C communication bus and power is supplied through the main power supply. The user cannot manufacture or add further nodes to the RDAS as access is not given to the user to extend the RDAS firmware or to add generic nodes and so the operation of the RDAS is limited to the use of the nodes that AED Electronics provides.

As mentioned above, the RDAS makes use of the I2C communication bus between its extension nodes. I2C is a two-wire communication bus including a clock line and a data line[11]. This bus includes bus arbitration and a standard 7-bit addressing system. However, the protocol does not include any data correction, such as cyclic redundancy checks and message retransmission. This makes using this bus to transmit event-critical data unsafe, especially as this bus does not make use of differential communication making it particularly susceptible to electromagnetic noise. In the case of the RDAS no event-critical data is transmitted over the I2C bus. However, the integrity of the data stored on the ground station is dependent on the integrity of the data on the I2C bus.

Telemetry Extension The telemetry extension for the RDAS operates over the 433MHz ISM band. It uses a Radiometrix transmitter-receiver pair and so operates only in a single direction; from the RDAS to the ground station. The RDAS transmits the data it measures through the telemetry module to the ground station for storage and display on a ground station computer. The display of the data is done in real-time; however

there is no way for the ground station to control the rocket in response to this data. With the correct antennas mounted on the ground station and the RDAS, the range of the RDAS telemetry can be extended to a maximum of approximately 15km. The telemetry extension provides a redundant means of data storage through the ground station.

The greatest limitation of the telemetry for the RDAS is that the user cannot be certain of the integrity of the measurement data. There is no error checking and correction of the packets sent to the ground station because the link is in only one direction. It should be noted however that the integrity of the data is not critical in the case of the RDAS because the telemetry data is not used to make any critical in-flight decisions.

GPS Extension The GPS extension for the RDAS provides GPS measurements through a commercial GPS Receiver. Such receivers adhere to ITAR regulations. This restricts the acceleration and velocity in which GPS receivers may return position information. This means that the GPS will only return position readings after the burn phase of the rocket motor when the acceleration drops below 4g and when the rocket is travelling below a velocity of 515m/s. Generally this means that the GPS receiver will only regain lock with the GPS satellite signal at apogee. This renders GPS measurement almost useless for position and altitude calculations in the first half of the flight. On the RDAS the data from the GPS is logged to the EEPROM and transmitted to the ground station.

Accelerometer Extension The accelerometer extension for the RDAS provides a further two axes of linear inertial measurement which may be chosen by the user through the orientation of the extension board. The data is logged by the main RDAS board to EEPROM and transmitted to the ground station.

Event Detection

Launch In order to detect launch the RDAS uses two simultaneous approaches for redundancy. Firstly, the RDAS provides the user with the ability to attach a break-

wire to the main circuit board. This feature is essentially a manual switch. When the rocket is on the launchpad, a jumper is connected across two pins creating a closed circuit. A wire is attached to the jumper and to the launchpad so that when the rocket launches the jumper is pulled off the two pins creating an open circuit. In this way launch is detected. Secondly, the RDAS is able to detect launch through a large spike in acceleration measured on its Z-axis accelerometer.

Apogee According to the RDAS manual, detection of apogee is done through two different methods. This redundancy is provided because the detection of apogee is vital for triggering the launch of recovery systems. The first method is a timer. The user, knowing the technical details of the rocket and the launch, should be able to calculate the time the rocket will take to reach apogee. The user may then input this time into the RDAS and the RDAS will trigger the recovery systems at the user-selected time. The second method the RDAS uses to detect apogee is through the use of the acceleration data from the Z-axis accelerometer. According to the RDAS manual, the RDAS integrates this acceleration data and when the integral calculated reaches zero, that is, when the rocket reaches a velocity of zero, apogee is detected and the recovery systems are triggered [6]. The problem with this approach is that the Z-axis accelerometer measures acceleration in the rocket's reference frame. As in figure 3.1 the Z-axis is the axis through the nose and tail of the rocket. This means that for flights that are not perfectly vertical the accelerometer is not only measuring acceleration in the vertical direction, that is in altitude direction, but also horizontally over the surface of the Earth. The zero velocity point will therefore not be a true calculation of apogee. This measurement is therefore a poor estimation of apogee that deteriorates the more horizontal a given rocket flight is. In fact apogee may not be detected at all in very horizontal flights because the velocity in the Z-axis of the rocket may not reach zero before the rocket hits the ground. This is because the Z-axis rotates with the rocket. For very horizontal flights the component of the Z-axis acceleration affected by gravity is small. In this case drag has a much larger affect on the acceleration of the rocket in the Z-axis. However, unless the opposing wind speed is very high, and therefore the relative velocity between the wind and the rocket is very high, it is unlikely that the resulting drag force will decelerate the rocket enough for

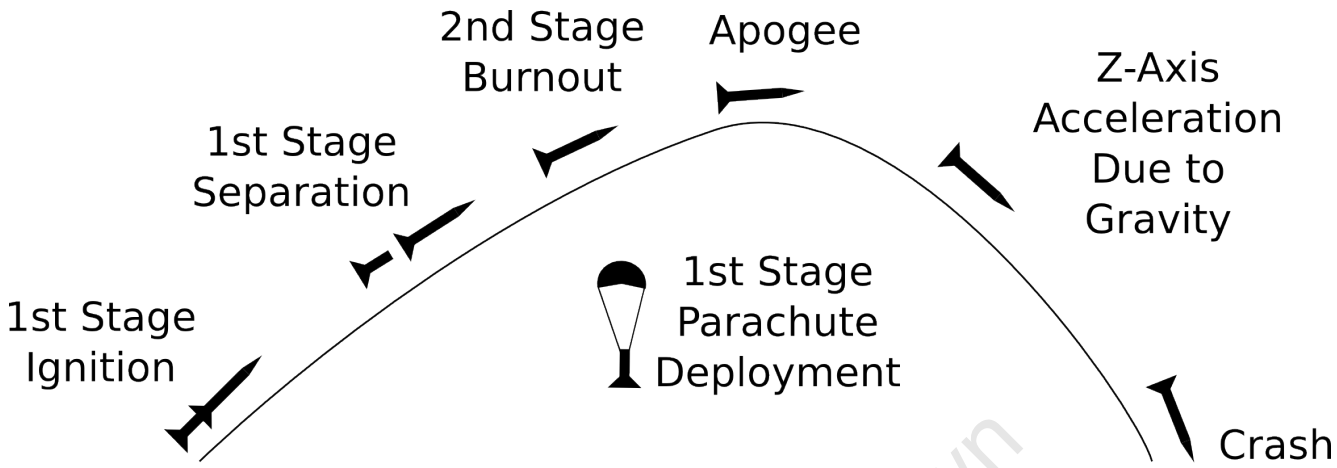

Figure 2.3: Trajectory scenario with a large initial launch angle.

the Z-axis velocity to become negative. Figure 2.3 illustrates a flight trajectory with a large horizontal component.

Graphical User Interface

The RDAS graphical user interface (GUI) provides the user with a real-time display of the data sent from the RDAS in the rocket, provided there is a working telemetry link between the ground station and the rocket. The program is able to display the data in real-time plots and to mark in-flight events such as launch and apogee on the plots for the user to see. It is also able to save and store data for later analysis. The program is able to plot the trajectory of the rocket for the user when using the GPS extension board. All the measured data may be accessed in raw form from the program.

The RDAS GUI allows the user to download the data stored on the RDAS EEPROM through the serial connection. The serial connection also allows the user to configure the RDAS, for example, to use the apogee detection timer. Finally the serial connection allows the user to calibrate the sensors on the RDAS through the GUI.

Lessons from the RDAS to be Carried Forward

The RDAS provides a minimum benchmark for a useful flight computer to measure the performance of a rocket in-flight and to support operational rocket flight through the detection and control of in-flight events. One of the major drawbacks of the RDAS is its inability to use its extensibility to increase its redundancy in event detection and control of rocket flight. So, for example, the user may include a further two accelerometers, a GPS and rotational inertial sensors through the digital or analogue inputs, but the RDAS is incapable of making use of this extra data to improve or extend its performance.

A number of important things have been learnt through the use and analysis of the RDAS. These include:

- The importance of redundancy and so redundant design in rocket flight computers.
- The importance of reconfigurability and extensibility in a reusable flight computer.
- The need for modularity to enable extensibility and reconfigurability and thus to overcome the drawbacks of the monolithic architecture.
- Numerous lessons pertaining to the operational limits of transducers and how they affect the operational limits of a flight computer in rocket applications.
- The importance of clear status indications when the flight computer is not on the work bench. In the RDAS case this is done through the buzzer.

2.3 Portland State Aerospace Society LV2b Flight Computer

The Portland State Aerospace Society (PSAS) at Portland State University is an education aerospace project involved in building state-of-the-art sounding rockets including

state of the art flight computers. This review will cover their LV2b flight computer[7].

LV2b Architecture

Figure 2.4 shows a diagram of the flight computer architecture of the PSAS LV2b flight computer.

As shown in Figure 2.4, the LV2b flight computer uses a master-slave architecture with a single master controlling a number of independent slaves. Each feature on the flight computer is given to a new node to make it completely independent of the rest of the flight computer.

Communication Bus The PSAS LV2b uses a USB hub to facilitate communication between the master node and the slave nodes. This has the advantage of providing very high bandwidth 12Mb/s whilst still providing the noise resistance of a differential bus. The draw back is that USB requires the master node to initiate all communication. So, for example, if the inertial measurement unit needs to transmit some data to the ground station, the master node must poll the inertial measurement unit for the data and then send it to the telemetry node. This means that if the master node fails during a flight all inter-node communication is stopped since USB hub is completely dependent on the master node.

The reasons PSAS decided to use USB for the inter node communication were largely based on software concerns. One of the major concerns raised by PSAS in their selection was that the use of CAN would limit their control of the sending of messages. CAN will retransmit a message until it is received. PSAS's point was that they would prefer to drop a message and receive the next critical packet then to delay a critical message. Further they were concerned that there was little open source support for CAN and that CAN is limited to 1Mb/s.

Based on the limitations of USB in terms of its dependency on a single node to function this writer does not consider USB a good design decision. If flexibility in the transmission protocol and higher bandwidth were required, a solution like RS - 485

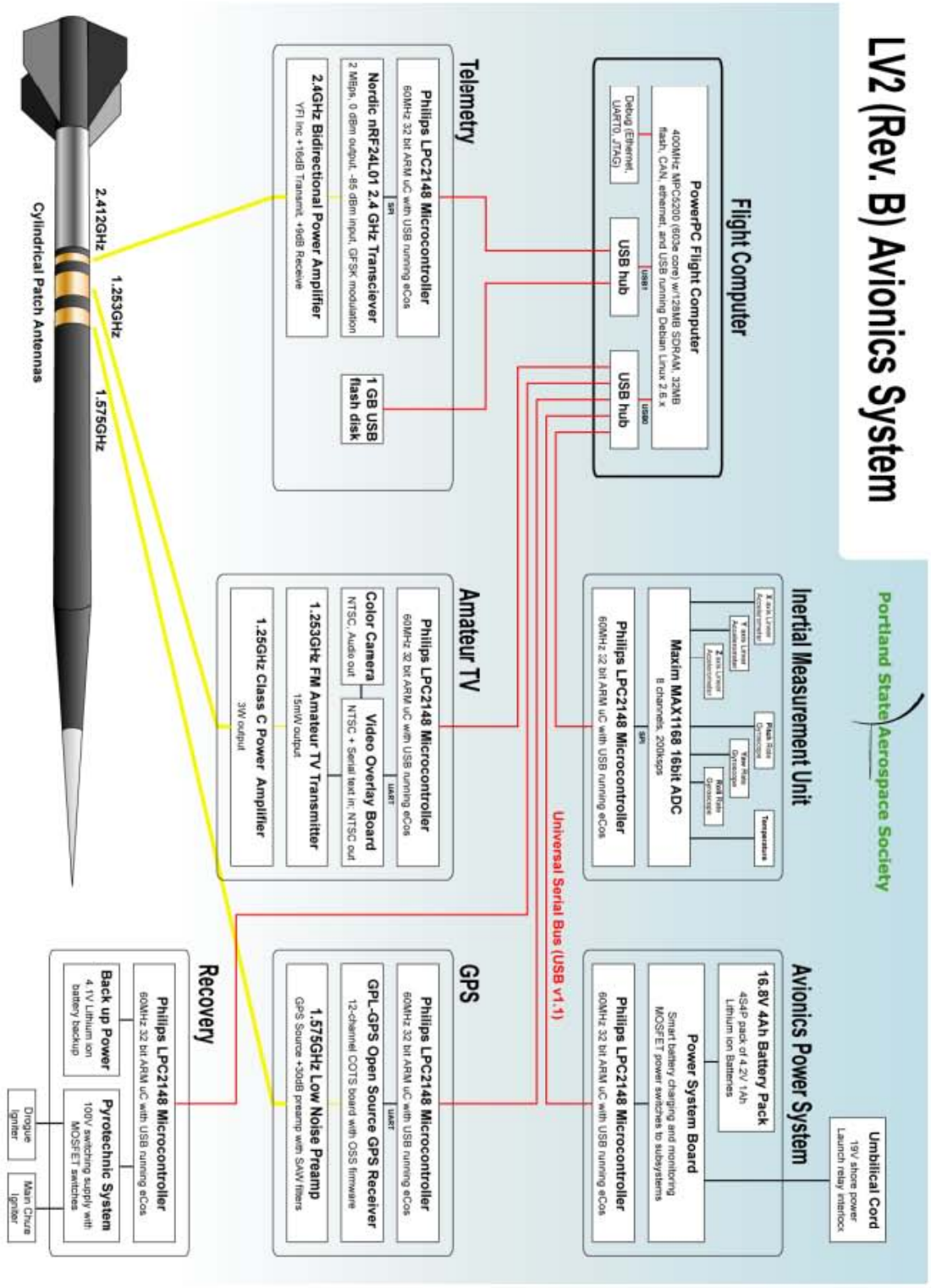

Figure 2.4: PSAS LV2b flight computer architecture [7].

would have been more suitable. RS-485 provides a differential bus like the CAN bus capable of data rates up to 35Mb/s. It does not force the user to use any specific transmission protocol and it does not depend on a single node to function correctly. Furthermore, this writer does not consider the existence of open source support as a good factor to consider when choosing a subsystem as critical as the communication bus.

Master Node The master node on the LV2b controls the USB hub, as well as being the connection point for external communications with a computer. The node is run by a Power PC running a real-time operating system (RTOS) called eCOS which includes built in support for USB communications.

Slave Nodes

Common Slave Node Blocks Each of the slave nodes has its own 32-bit 60MHz ARM processor controlling communications with the master node and facilitating the functions of the node. The major requirement that PSAS had for this processor was that it should be capable of over 10MIPS and that it should support USB.

Power Management System The PSAS power management unit uses a switch mode power supply to regulate their 16.8V battery supply down to 3.3V. The battery voltage is regulated down to 3.3V through the use of a switch mode buck regulator. A secondary 5V supply is made available through a Low Drop Out (LDO) regulator for components requiring the high supply voltage. Supply of power to the individual nodes is controlled through a set of mosfet switches.

Inertial Measurement Unit The PSAS inertial measurement unit provides measurement of inertial data in 3 linear axes, X, Y and Z and 3 rotation axes, pitch yaw and roll. It also provides pressure measurement and temperature measurement. The temperature measurement is used to calibrate the MEMs sensors. This is because the

output of MEMs sensors drifts with temperature, this drift can be characterised and so the outputs of the MEMs sensors can be calibrated in real-time with temperature.

Recovery The PSAS recovery system provides igniters for the main parachute and the backup parachute. The channels are controlled through the use of mosfet switches. Further backup power is supplied to the node in case of a flight computer failure. This helps to prevent a flight failure in the event of a major flight computer failure.

Telemetry The PSAS telemetry system makes use of a Nordic nRF24L01 transceiver to facilitate bi-directional half duplex communication. The Nordic receiver includes a robust transmission protocol capable of error detection and retransmission of data at a maximum data rate of 2Mb/s. In order to extend the range of the 0dBm transmitter a bi-directional 2.4GHz amplifier is attached to the output of the transceiver. It should be noted that bi-directional amplifiers are particularly costly.

The telemetry system also includes the persistent storage of the PSAS flight computer although it is connected to the main node via a separate USB connection. Persistent storage is facilitated through the use of a USB flash disk.

System and Firmware

As mentioned above the LV2b uses a real-time Linux operating system called eCOS on its main node. This is a questionable design choice, given that it is necessary for the flight computer to be reliable for mission success. The eCOS Linux kernel makes use of a pre-emptive task scheduler [12]. A pre-emptive scheduler entails each task being assigned a priority. Tasks with a higher priority are then able to pre-empt tasks of a lower priority, that is, they are able to take over the CPU from the lower priority task. If this type of scheduler architecture is not carefully designed and thoroughly tested it can lead to a failure called a priority inversion. This entails a low priority task locking a shared a resource with a mutex and then blocking a higher priority task from pre-empting CPU and utilising that resource. A mutex, or mutual exclusion object, is a method of allowing two program threads to share one resource. When one thread

uses a resource it locks the mutex to indicate that the resource is currently in use, thus blocking other threads from using the resource. One example of such a failure was on NASA's Mars Pathfinder mission. In this case a low priority task locked a shared resource with a mutex. In doing so it blocked a higher priority task that required the resource. The low priority task was then pre-empted by a medium priority task that did not require access to the shared resource. This delayed the low priority task from completing, which in turn delayed the higher priority task. The watchdog timer then detected that the high priority task had not been executed which led to a full system reset [13].

The major issue with this scheduler architecture is simply that it requires a huge amount of testing to be sure that all possible task combinations do not cause priority inversion or deadlock. In large complex systems it may be impossible to test the system comprehensively [9]. It should be noted that there are a number of complex solutions to the priority inversion problem, including for example using priority inheritance. This involves assigning the low priority task, that is locking the shared resource, the highest possible waiting priority so that medium priority tasks are unable to pre-empt it. The primary issue with these solutions is that it is still difficult to test every combination of tasks. It is also impossible to guarantee a precise execution time for a task without elevating its priority above all other tasks.

A further potential problem in badly designed or insufficiently tested pre-emptive architectures is process deadlock. This may occur for example in a circular wait situation. This entails a set of processes each waiting for the next process to release a shared resource. So, for example, for the set of processes P1, P2, P3 if P2 is waiting for P1 to release a resource, and P3 is waiting for P2, and P1 is waiting for P3 the scheduler will be in a deadlocked state and unable to continue.

It should be apparent to the reader that in order to manage task priorities and to detect or prevent deadlock situations a number of algorithms are required to run along-side the scheduler. This significantly increases both code size and complexity [9]. This increase in code size and complexity makes it more difficult to test properly a system to ensure its reliability.

Lessons Learnt for the PSAS LV2b

The LV2b is substantially more complex and more capable than the RDAS flight computer. A number of lessons have been learnt in analysis of the LV2b flight computer. These include:

1. The ability of the master-slave architecture to achieve extensibility and modularity.
2. The review of the communication buses in question has informed the selection of possible communication buses for the SAAO flight computer.
3. The self calibration of the inertial measurement unit through the temperature sensor was noted as something to be included in the SAAO flight computer.
4. The use of a backup battery pack for the recovery system was noted as something to be used in the SAAO. This is due to the fact that it helps to avoid a flight failure in the case of a major failure on the flight computer.
5. The importance of the use of a co-operative scheduler in the design of reliable time-critical embedded systems.

2.4 Conclusion

In summary of this chapter; we have analysed two projects of similar intent, the RDAS flight computer and the PSAS LVb2 flight computer. Each flight computer has been analysed for its strengths and weaknesses and from this analysis lessons to be taken forward in the design of the SAAO flight computer have been noted. In the next chapter we will analyse the requirements and constraints for the flight computer provided by SAAO, breaking them down into the functions required. We will also present the testing required to verify the operation of the flight computer.

Chapter 3

SAAO Requirements and Functional Analysis

3.1 Introduction

The following chapter presents the requirements for the SAAO flight computer provided by the South African Astronomical Observatory. These requirements are analysed and translated into the functions that are required of the flight computer. Following the presentation of the functions required of the flight computer, the testing procedures are given. These test procedures are required to verify that each of the functions has been successfully implemented.

3.2 SAAO Requirements

What follows are the requirements given by SAAO for the design of the flight computer.

The Purpose of the Flight Computer

The purpose of the flight computer is to measure the performance of a sounding rocket and to control key events all phases of flight. This includes measuring and recording atmospheric and inertial data in-flight. Through the processing of this data the flight computer detects in-flight events, such as apogee, and responds with the appropriate action, such us the deployment of parachutes. The flight computer must have the capacity to support additional payloads on the rocket, for example experiments.

Performance Requirements

Rocket Performance Measurements

R1. The flight computer should be able to take inertial measurements in 6 degrees of freedom. It should also be able to measure atmospheric variables relevant to the performance of the rocket, such as temperature and pressure. All of this information should be used to determine the position of the rocket.

Detection of In-Flight Events

R2. The flight computer should be able to detect the following in-flight events:

- R2.1 - Launch
- R2.2 - Stage Burnout
- R2.3 - Separation
- R2.4 - Stage-Ignition
- R2.5 - Apogee
- R2.6 - Landing

Ground Services and Status Indications

R3. It should be possible to communicate with and control the flight computer on the launch pad.

R4. It should be possible to power the flight computer from an external power source on the launch pad so that the battery life of the flight computer is not used up during pre-launch preparations.

R5. The flight computer should give auditory feedback relating to its status on the launch pad. Status information should also be continuously communicated to the ground-station.

R6. The ground station should include a computer with the ability to send commands to the flight computer and to receive data from the flight computer. Further, this computer should be able to process, store and display this data in real time.

Flight Computer Systems

R7. The flight computer should be able to manage and provide power to all its subsystems.

R8. The flight computer should be able to store all the measured flight data on board for later retrieval.

R9. The flight computer should be capable of triggering a minimum of five igniters in response to in-flight events.

R10. The flight computer should include a telemetry capability to allow for real-time data transmission to the ground station and for ground station control of the flight computer.

R11. The flight computer should include an independent flight termination capability with a manual override. This is to prevent the rocket from crashing in civilian areas and causing damage to persons or property.

Flight Computer Architecture

R12. The flight computer should allow for expansion and the inclusion of mission-specific payloads.

Stresses and Constraints

R13. The flight computer should be able to perform its functions within the following parameters and constraints:

- R13.1 - Size: The flight computer should fit into an electronics bay of a minimum diameter of 60mm and a maximum length of 300mm.
- R13.2 - Mass: The flight computer should not exceed a mass of 500g including its batteries.
- R13.3 - Power Usage: The flight computer should have enough battery life to last a minimum of 100 minutes in nominal operating conditions. It should be possible to switch out battery packs without disturbing the operation of the flight computer.
- R13.4 - Acceleration: The flight computer should be able to operate up to a maximum of $\pm$50g acceleration.
- R13.5 - Temperature: The flight computer should be able to operate down to a minimum temperature of -50°C and a maximum of 80°C.
- R13.6 - The flight computer must be electrically isolated from its surroundings or grounded to the rocket
- R13.7 - The flight computer should be mechanically robust enough to survive multiple installations and removals from the rocket payload bay.
- R13.8 - Noise Immunity: The flight computer should be designed to withstand ambient electromagnetic radiation. The value of ambient electromagnetic radi-

ation is difficult to decide upon given that it will fluctuate greatly, for example, with proximity to cell phone towers.

Reliability and Safety Requirements

Robust Design

R14. The flight computer should be designed in such a way that a failure in one subsystem does not cause a failure of the flight computer as a whole.

R15. The flight computer should be able to recover from program hardware or firmware faults during flight to prevent mission failure.

System Redundancy

R16. Mission-critical subsystems should have some form of redundancy to prevent them from becoming potential single-point failures.

Reliability

R17. The flight computer should detect and record flight events with a reliability of above 95%.

R18. The flight computer should include error detection and correction capabilities in its communication and data processing hardware or firmware.

Life Cycle Management Requirements

R19. Wherever possible standard components (COTS) and communication protocols should be used to ensure that the lifespan of the flight computer is not cut short by an obscure component or protocol falling out of use.

R20. The flight computer should be modular and easily extensible in such a way as to improve the lifespan of the flight computer.

R21. Finally the design of the flight computer must make it easy for future users to update and change the firmware to suit their requirements.

Environmental Requirements

R22. Due to the inherently risky nature of rocket flight and the high probability that the flight computer would be destroyed in a crash, the components used to construct the flight computer should be Restriction of Hazardous Substances Directive (ROHS) compliant wherever possible. This will prevent toxic substances from causing environmental damage in the case of a rocket crash.

3.3 Functional Analysis

This section will break down SAAO's flight computer requirements into the required flight computer functions. This will aid in the production of a clear specification to inform all further design and testing. Each function is linked to one or more requirements.

3.4 In-Flight Functions

In-Flight Measurements

The following set of functions relate to the measurement of performance and atmospheric variables during flight. For reference with the following functions, diagram 3.1 shows the orientation of the axes with respect to the rocket.

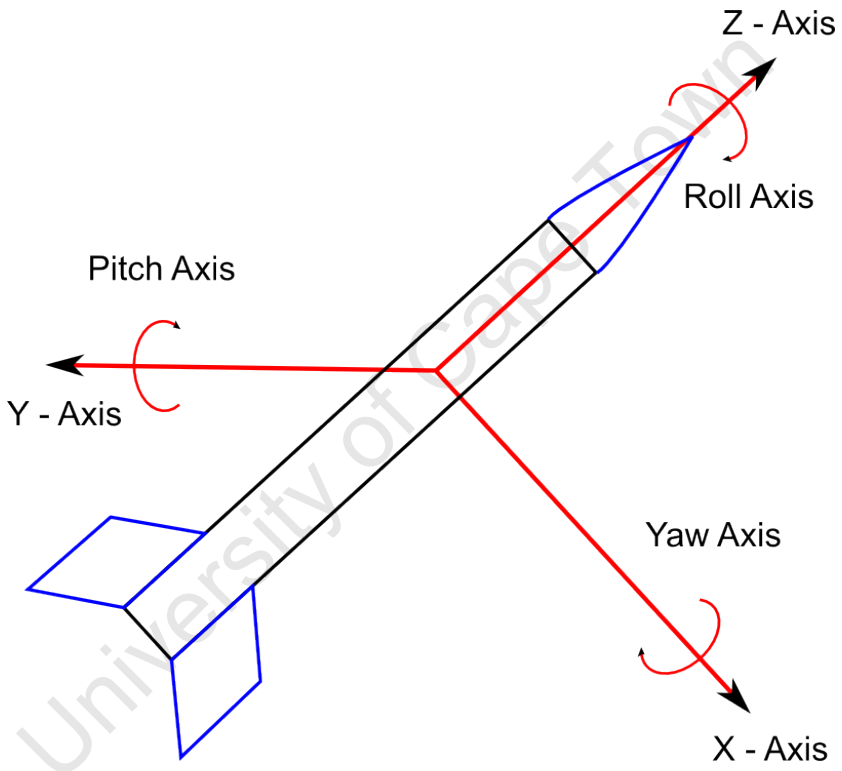

Figure 3.1: Diagram showing the orientation of the axes with respect to the rocket.

Inertial Measurement

Function Reference	Function Description	Requirement Reference
F1	The flight computer must be able to measure acceleration in 3 linear axes; X, Y and Z and 3 rotational axes; pitch, yaw and roll. This data should be used calculate velocity and position. The linear axes should have a maximum acceleration measurement range of $\pm$200g in the Z-axis and $\pm$20g in the X and Y axes. The rotational axes should have a minimum measurement range of $300^{\circ}/s$ The measurement rate should be a minimum of 200Hz.	R1

Pressure Measurement

F2	The flight computer should be able to measure pressure relative to its launch point. This will provide a second means of calculating altitude. The rate at which the flight computer can measure pressure should be 200Hz.	R1

Position Measurement

F3	The flight computer should be able to take GPS readings, preferably differential through the use of a GPS receiver both at the ground station and on the flight computer. This will provide a second measure of position to null the cumulative error in the inertial position calculations.	R1

Temperature Measurement

F4	The sensor is primarily to monitor the conditions in which the flight computer is operating; therefore it is not necessary to link the sensor to the outside atmosphere.	R1

Power Measurement

F5	The flight computer should be able to monitor the current draw of each of its subsystems as well as the voltage of the batteries. These measurements should be at a minimum of 10Hz. This information will allow the flight computer to enter low-power modes in the case of low battery voltages. It will also allow the flight computer to detect faulty subsystems when they use too much power.	R5, R7

State and Functionality Monitoring

F6	The flight computer should be able to monitor whether or not each of its subsystems are functioning correctly. The state information should be made available at the ground station. In the case of the recovery system the status of the igniters should be monitored, that is, whether the igniters are open or closed circuit.	R5, R7, R15

In-Flight Data Storage

F7	All of F1 - F6 should be stored in the flight computer's on board non-volatile memory. It must be fast enough to store data in real time. The flight computer will probably produce between 10 - 80kb/s of data, therefore for enough capacity to store 100 minutes, a minimum of between 8 and 64 MB of storage is required.	R8, R16

In-Flight Data Transmission

F8	All of F1 - F6 should be transmitted by the rocket to the ground station. The flight computer should be able to send and receive data. The data link should include error checking and the capacity for the correction of errors. The link should be fast enough to transmit all the required data and to resend messages that are corrupted. A range of 50km and data rate of a minimum of (100kb/s) is expected.	R3, R10

3.5 Events

The following functions relate to the detection and initiation of events during flight.

Event Detection

Launch Detection

F9	The flight computer should be able to detect the beginning of a flight so that it can start storing and transmitting flight data. Launch may be detected through means of a break-wire, a large, sudden change in acceleration, a large change in initial pressure or a user override through the telemetry function.	R2.1

Stage Burnout Detection

F10	The flight computer should be able to detect when a motor runs out of fuel so that the stage may be separated from the rocket and a new rocket motor stage ignited. This event will entail a negative acceleration in the Z-axis due to drag and gravity. It may therefore be detected through inertial measurement. A manual override through the telemetry system in the case of a failed detection should also be possible.	R2.2

Separation Detection

F11	After detecting stage burnout the flight computer should be able to trigger a separation with the burnt out motor stage. This event's completion should be confirmed before a new motor stage is ignited since premature ignition would cause a major failure. Separation could be detected via a break-wire or via manual override through the telemetry system in the case of a failed detection.	R2.3

Re-Ignition

F12	The flight computer should be able to detect when a new rocket stage is ignited. This may be detected by a sudden increase in acceleration or a sudden increase in the rate of decrease of pressure and or temperature.	R2.4

Apogee Detection

F13	The flight computer should be able to detect when the rocket reaches the apex of its flight so that it can trigger the recovery systems. Detection may be through an increase in pressure measurement, through the use of a timer system, an increase in acceleration in the Earth's negative Z-axis or zero velocity in the Earth's Z-axis or by manual override through the telemetry system in the case of a failed detection.	R2.5

Landing Detection

F14	The flight computer needs to be able to detect the landing of the rocket. This may be done by measuring zero velocity, a static pressure measurement approximately the same as the launch pressure, or via a large acceleration in the positive Z-axis. Once detected the flight computer should turn on a tracking beacon.	R2.6

Dangerous Trajectory Detection

F15	The flight computer should be able to detect a trajectory that is outside a predetermined set of safe trajectories. One method of doing this would be to compare the horizontal displacement of the rocket with some predetermined radius outside of which the rocket's trajectory would be considered dangerous. This should also include the ability to manually set the trajectory as dangerous through the telemetry system. This information would be passed to a flight-termination module.	R11

Event Initiation

Launch Initiation

F16	The flight computer should be able to fire a launch igniter at the command of the ground station. The igniter driver circuit should be able to provide 3A for a duration of 0.5s. Further, hardware and software locks are required to ensure that the igniter does not fire due to a fault or false detection.	R9

Stage Separation

F17	Having detected stage burnout, the flight computer should be able to fire a stage separation charge igniter to separate the burnt out stage from the rocket. The igniter driver circuit should be able to provide 3A for a duration of 0.5s. Further, hardware and software locks are required to ensure that the igniter does not fire due to a fault or false detection.	R2.2, R9

New Stage Ignition

F18	The flight computer should be able to ignite each subsequent stage of the rocket. The igniter driver circuit for each stage should be able to provide 3A for a duration 0.5s. Further, hardware and software locks are required to ensure that the igniter does not fire due to a fault or false detection.	R2.3, R9

3.6 Ground Station

Telemetry Data Display

F19	The ground station should be able to display data received from the rocket in real time, before and during flight. This should be through a ground station graphical user interface connected to the ground station telemetry. The data display should include data processing to ensure the data displayed is in a format the user can understand. Further, there should be a graphical illustration of the trajectory of the rocket in comparison to some simulated expected trajectory to provide the user with clear information to react to in the case of a dangerous trajectory.	R6, R10

In-Flight Commands

F20	The ground station should provide the user with the ability to send commands to the flight computer via the telemetry link. This is to provide the user with the ability to react with manual overrides in the case of failed detections or to initiate a destruction command in the case of a dangerous trajectory.	R6, R10

Auditory Feedback on Launchpad

F21	When on the ground the flight computer should provide an audible indication of its status to provide the user with clear information with regards to its operational status.	R5

Table 3.1: Functional analysis with reference to SAAO requirements.

3.7 Testing

In order to meet the requirements of SAAO and to ensure that the flight computer will operate under the conditions of rocket flight given by SAAO, the flight computer shall be tested using the process outlined in this section of the specification. The following test categories are derived from NASA Dynamic Environmental Criteria handbook [14].

Test Categories

Qualification Tests

According to the NASA Dynamic Environmental Criteria handbook the purpose of a qualification test is "...to demonstrate with margin that a hardware design is adequate to perform as are required throughout the mission (pre-flight and flight) environmental exposures." [14]

Failure of these tests is incurred if and only if the failure is the result of a design deficiency. Failures that are due for example to workmanship error are not relevant here. The goal is to verify that the design properly meets the specification with margin.

Acceptance Tests

According to the NASA Dynamic Environmental Criteria acceptance tests are carried out on hardware that has passed a set of qualification tests. The purpose of these tests is to "...detect workmanship errors and/or material defects in the manufacture and assembly of the hardware, and to demonstrate that the hardware is representative of the qualified design."

Failure here is only incurred if the failure is due to a workmanship error or a material defect. Due to the fact that the hardware has undergone and passed qualification tests there should be no design faults.

Protoflight Tests

A protoflight test is a test applied to one-of-a-kind flight test hardware in order to meet the goals of both qualification and acceptance testing.

A failure may be due to both design deficiencies and workmanship or material deficiencies. In the case of workmanship and material deficiencies the fault should be repaired if possible and the test continued.

Tests to Be Performed

Qualification Tests

Design Verification Tests In order to validate that the design of the hardware functions according to the requirements in this specification the flight computer should be placed in an environment where flight conditions are simulated. The performance of the flight computer should be measured and recorded for analysis.

The following process should be followed for validation:

1. First, the flight computer should be tested in ideal conditions. This should be done to verify the functionality of the flight computer design. For example

the ability of the communication bus to handle all the data expected during a flight. This may be achieved through the use of In-the-Loop testing. In-the-Loop testing involves modelling the expected inputs to the flight computer in order to simulate a flight. The modelled inputs are then put through the flight computer and the response of the flight computer is recorded. The performance of the flight computer in this simulated flight should be measured and analysed to determine if the design meets the specification.

2. Once the design has been verified in the ideal case the ideal model should be altered to include measurement noise and non-ideal trajectories. The limits of the design to withstand noise and non-ideal flight conditions should be characterised.

3. Once this has been achieved the above tests should be rerun under the environmental conditions expected during flight. Flight simulation through In-the-Loop testing shall be carried out along with the following tests derived from MIL-STD-1540C [15]:

- Temperature stress testing should be carried out by subjecting the flight computer to the temperature shock to be expected during flight. A change of up to 70°C over 60 seconds is to be expected. The performance of the flight computer should also be tested at the extremes of its operational temperature. That is down to -50°C and up to 85°C. In-the-Loop simulations should be run multiple times to characterise fully the performance at these extremes.
- Vibration testing should be carried out to ensure the flight computer can operate over long periods of time under the expected vibration from the rocket motor. In-the-Loop simulation testing should be carried out in a vibration chamber multiple times to ensure operation in these conditions.
- Acceleration testing should be carried out to ensure the flight computer can operate in a sustained acceleration environment of up to 50g over prolonged periods longer then the expected burn phases of the rocket motor.
- Shock testing should be carried out to ensure that the flight computer can operate through a shock of up to 200g in the Z axis.

- Vacuum testing should be carried out to ensure that the flight computer's electronic components do not out-gas. The vacuum should be lower pressure then 0.1kPa, which is the atmospheric pressure at 50km.

Acceptance Tests

To find workmanship defects, voltage level tests, continuity tests, and visual inspection should be carried out on each flight computer. Workmanship faults may also be discovered by simulation testing each flight computer in the ideal case and verifying each individual function.

Protoflight Tests

Once the flight computer has been qualified, a number of flight tests should be carried out to ensure the flight computer's operation in the real world environment. In each case the flight computer should be paired with a second commercial flight computer which should be responsible for the control of the rocket. The flight computer should be tested to verify correct operation and correct detection of events in flight. The data should be analysed and compared to the data from the commercial flight computer to verify the operation of the flight computer.

Once several successful flights have been undertaken the flight computer should be placed in a rocket on its own. This final validation of the operation of the flight computer should only be done once the tester is certain of the operation of the flight computer based on the previous testing.

Testing Procedures

The following section details the procedures for verifying that the flight computer operates within the constraints set out by SAAO in requirement R13. It also includes the procedures to verify the operation of the flight computer outlined in functions F1 to F21.

R13.1 - Size The flight computer should be able to fit into a 60mm diameter cylinder with a maximum length of 300mm. Therefore to verify this constraint the flight computer should be mounted within a 60 mm diameter cyclinder.

R13.2 - Mass In order to verify the mass of the flight computer and its batteries, the flight computer and its batteries should be weighed using a standard laboratory scale.

R13.3 - Power Usage In order to verify that the flight computer is able to operate nominally for 100 minutes, the flight computer should be run on a bench for a 100 minute period. The operation of the flight computer should be logged for the duration of the test to ensure that it continuous to operate nominally for the full 100 minutes.

R13.4 - Acceleration In order to verify the flight computer's ability to operate through large acceleration spikes, the flight computer should be subjected to acceleration shocks. This may be in the form of drop testing. The flight computer's nominal operation should be verified before and after the test.

R13.5 - Temperature In order to verify the operation of the flight computer within the temperature limits set out by SAAO, the flight computer should be placed in a temperature chamber capable of producing the temperatures set out in R13.5. The flight computer should be monitored by logging its operation. This log must then be analysed to ensure the flight computer operated nominally over the duration of the test.

The flight computer should also be subjected to temperature shock, that is rapid changes in temperature from 25°C down to the minimum operation temperature of -80°C over a period of one minute. The temperature should then by increased back to 25°C over a period of 5 minutes. This is to simulate the change in temperature over the duration of a rocket flight. The flight computer should be monitored to ensure nominal operation over the duration of the test.

R13.8 - Noise Immunity In order to verify the flight computer's immunity to noise, the flight computer should be monitored for nominal operation in an area with high levels of electro-magnetic radiation. One potential test location could be next to a cell phone sub-station. The flight computer should be tested over a duration of 100 minutes.

Another likely source of electro-magnetic radiation is from tracking radars at a test range. One potential method of simulating the electro-magnetic radiation from a radar would be to attach a directional antenna to a signal generator and to aim it at the flight computer. The test could be run in different frequency bands and with varying power levels to test the effect of the electro magnetic radiation on the flight computer.

R17 - Reliability In order to ensure that the event detection algorithm is able to detect flight events with a reliability of above 95%, the algorithm should be tested using datasets from actual rocket flights.

Further Environmental Testing The flight computer should be tested for nominal operation in a vibration chamber. The vibration test should be operated for up to 30 seconds at a time. This duration is well above the normal burn time of a solid rocket motor.

F1 - Inertial Measurement F1 states that the flight computer is required to have inertial measurement in six degrees of freedom. In order to test this, the flight computer should be subjected to a known acceleration in each axis individually. The measurement should then be compared to the known acceleration to ascertain whether or not the inertial measurement is operating correctly.

F2 - Pressure Measurement F2 states that the flight computer is required to be able to measure pressure. In order to test this, the flight computer should be tested over a range of known pressures. The measurements from the on-board pressure sensor should be compared to the known pressure in order to ascertain whether or not the pressure sensor is calibrated and operating correctly.

F3 - Position Measurement F3 states that the flight computer is required to be able to measure the position of the rocket through both GPS and inertial means. Firstly, the on-board GPS system should be tested for its ability to lock with GPS satellites in an environment similar to that of a launch site. Once lock has been verified, the position read-out of the GPS system should be compared to that of a known position in order to ascertain whether or not the GPS system is operating correctly.

Secondly, position derived from inertial measurement should be tested. Before these tests can begin the sensors should be characterised and calibrated. The flight computer should be displaced a known distance without any rotational displacement. The output of the inertial measurement unit should then be compared to the known displacement to ascertain whether or not position has been correctly calculated. Once linear displacement calculations have been verified, a rotational displacement should be included in the known displacement. Once again the output should be compared to the known displacement to ascertain whether or not the inertial measurement unit's output is correct.

F4 - Temperature Measurement F4 states that the flight computer is required to be able to measure temperature. In order to test this the flight computer's ability to measure temperature, the flight computer should be placed in a temperature chamber at a known temperature. The output of the on-board temperature sensor should then be compared to the known temperature to ascertain whether or not the sensor is properly calibrated and operating correctly. This test should be repeated over the whole range of temperatures that R13 requires the flight computer to operate in, with a resolution of $10°C$.

F5 - Power Measurement F5 states that the flight computer is required to measure both the current consumption of the individual nodes as well as the voltage levels of the batteries. In order to test the measurement of current consumption the flight computer's power channels should be connected to dummy loads to ensure the consumption is stable. The current through each node should then be measured with a multimeter. This known value should then be compared to the output of the current

measurement to ascertain whether or not the on-board sensors are operating correctly.

In order to test the correct operation of the voltage measurement, the flight computer should be connected to batteries. The voltage of the batteries should be measured using a multimeter. The known voltage should then be compared to the measured voltage to ascertain whether or not the on-board sensors are operating correctly.

F6 and F21 - Operation Monitoring and Status Indication F6 and F21 require that the flight computer is able to monitor its status, that is, whether or not each of the nodes are operating correctly. It is also required to indicate its current status audibly. In order to test this, the flight computer should be operated normally to ascertain whether or not it is able to detect and indicate normal operation. A node should then be disconnected from the CAN bus, and the audible status should be monitored to ascertain whether or not there is a change in the status notification indicating a malfunctioning node.

F7 - Data Storage and Transmission F7 states that the flight computer is required to be able to store all measured data, on-board, in non-volatile memory. This stored data should also be transmitted via telemetry to a ground-station computer.

In order to verify the storage of data in the on-board non-volatile memory, a known data sequence should be stored in the memory through the flight computer. This data should then be read out through the flight computer. The read-out data should be compared to the input data in order to ascertain whether or not the data is correctly stored in the non-volatile memory.

This same data known data set should be transmitted over the telemetry link. The transmitted data should be compared to the received data on the ground-station side in order to ascertain whether or not the telemetry link is operating correctly.

R17 and F9 to F15 - Event Detection Functions F9 to F15 as well as Requirement R17 state that the flight computer should be able to detect in-flight events. In order to test this, real flight data with known flight events should be run through the flight

computer's detection algorithms. The detection of in-flight events by the detection algorithms should be compared to the known events in the flight data.

F16 to F18 - Event Initiation F16 to F18 state that the flight computer is required to be able to initiate events in response to detecting events. In order to test its ability to initiate events, the flight computer should be connected to dummy igniters. The dummy igniters should include a resistive load capable of indicating that enough current is being provided. One example of this would be to use an incandescent light bulb. In this scenario, if enough current is provided, the light bulb would turn on. The event triggering and firing mechanisms should be tested with the dummy load and the load should be monitored to ascertain whether or not the mechanisms operate correctly.

F19 - Data Display F19 states that the flight computer system is to include a ground-station graphical user interface, which displays the flight computer's data in real-time. In order to test the GUI's ability to display data, the flight computer should be connected to the GUI and operated normally. The graphical user interface should then be monitored for its ability to display the data.

F20 - In-Flight Commands F20 states that the ground station GUI is required to be able to transmit commands to the flight computer. In order to test this the flight computer should be connected to the ground station GUI. Commands should then be sent from the GUI to the flight computer. The flight computer should then be monitored for a response to the given command to ascertain whether or not this function operates correctly.

3.8 Conclusion

In summary of this chapter, we have presented the SAAO requirements for a rocket flight computer. The SAAO requirements were then analysed by breaking them down into the functions they require. We then outlined the testing required to verify that

the flight computer meets the requirements given by SAAO. In the following chapter we will present the design of the SAAO flight computer based on the requirements, constraints and functions derived in this chapter.

Chapter 4

Design

4.1 Introduction

The following chapter presents the design of the SAAO flight computer. Each design choice is referenced back to the requirement or function that necessitated the design feature. The chapter begins with the selection of the architecture of the flight computer. This is followed by the detailed design of the flight computer based on the architecture selected. This includes, the selection of and design of hardware, the design of the operating system and the design of the algorithms required to process the data.

4.2 Flight Computer Architecture Design

The choice of architecture for this flight computer is critical to facilitate the modularity and extensibility requirements of the user, namely the requirements R12 and R20. What follows is a review of the architectures considered for this project.

Architecture Review and Selection

Monolithic Architecture

As shown in Figure 4.1, this architecture makes use of a single processor capable of controlling all the features of the system. The advantage of this architecture is that it requires the smallest number of components and the smallest amount of circuit board space of all the available architectures. Both of these are important points to consider bearing in the mind the budget and space constraints of the project as per R13. The biggest limitation of this architecture is its inflexibility with regards to the extensibility of the system. Further, redundancy is problematic given that the single processor is a single-point failure for the system. If the main processor fails all the functions of the system fail. When considering the modularity and extensibility requirements of SAAO, the limitations of this architecture render it incapable of fulfilling the requirements.

Interconnected Systems Architecture - Minimum Microcontrollers

A slightly altered version of the monolithic architecture is shown in Figure 4.2. It reduces the dependence of the system on a single processor. The improvement of this architecture over the monolithic architecture is the elimination of a single-point of failure on the circuit board. Functions are distributed over as many processors as are required. However, the functions are not completely independent of each other, so a failure in one processor would still not have a minimal affect on the system. The major drawback of this approach is still the lack of complete modularity. Although the functions are separated over a number of processors, there is still no standard connection to the system and so modularity and extensibility are not achieved in this architecture.

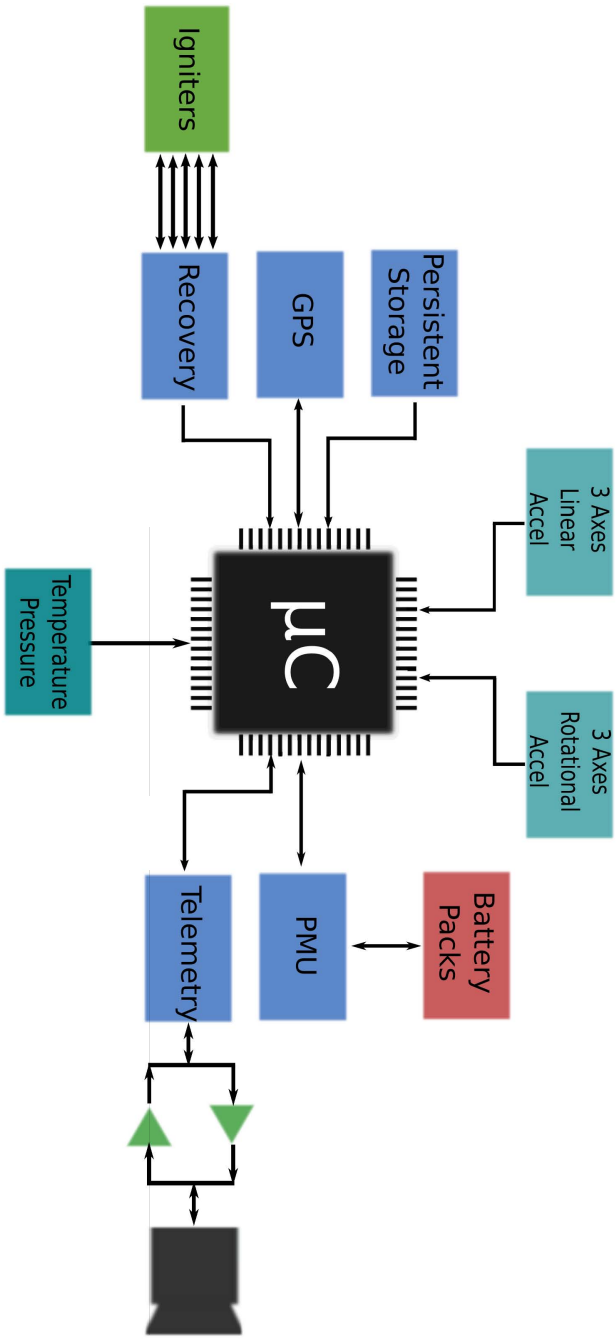

Figure 4.1: Monolithic architecture.

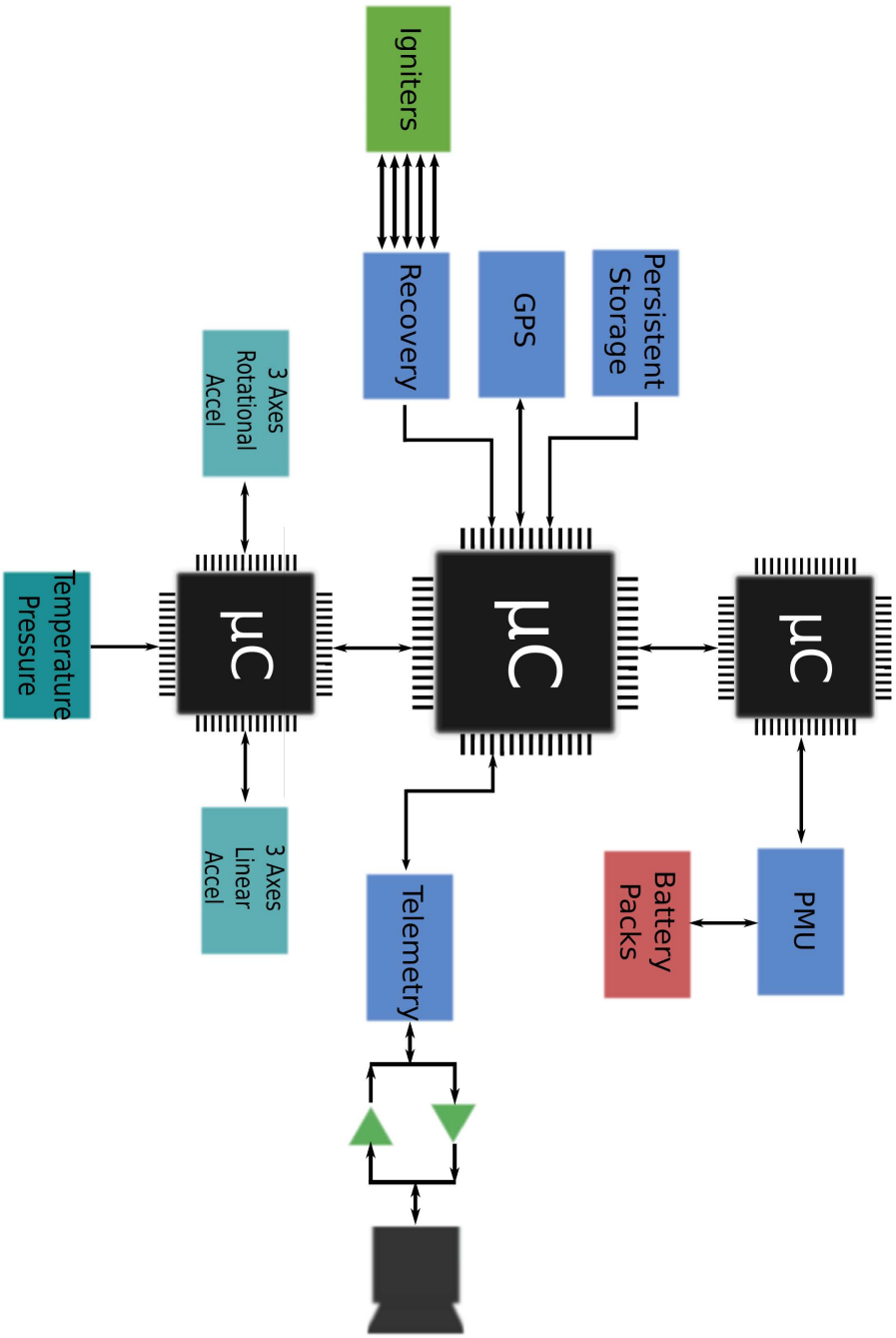

Figure 4.2: Interconnected architecture.

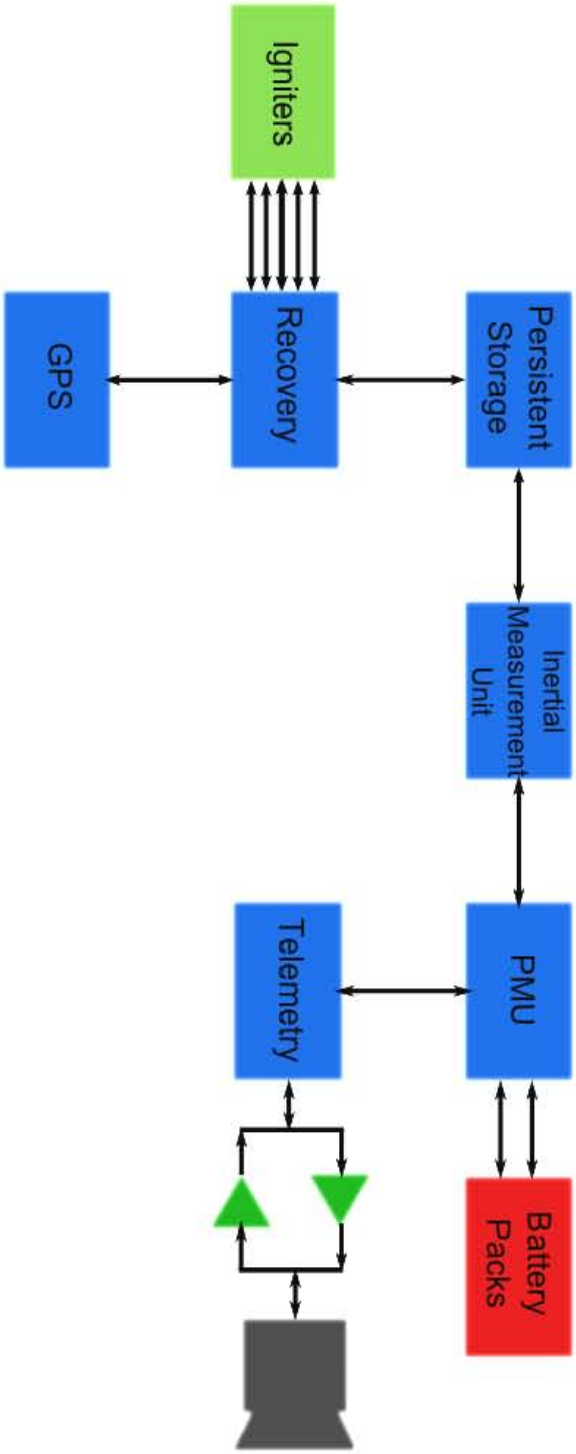

Figure 4.3: Distributed architecture.

Distributed Architecture - No Master

The distributed architecture shown in Figure 4.3 achieves complete modularity by distributing each flight computer function to an independent node. There is no master controlling node; each node carries out its function independent of the other nodes. Each node transmits and receives only the data relevant to its operation. The advantage of this architecture is that failure of one node has a minimum effect on the other nodes. Also, it becomes easy to add new functions to the system as a new node need only use the same power and communication protocol as provided to the other nodes. The major drawback of this architecture is that there is no control of the individual nodes in response to system status or events as outlined in F6. For example, if one node were to start to malfunction, there is no other node that could turn off the broken node. A second drawback is the large component count and the large amount of circuit board space required for this architecture.

Master-Slave Architecture

The master-slave architecture as shown in Figure 4.4, incorporates the modularity of the distributed architecture with the control of the monolithic architecture. In this architecture we have a single master node monitoring and controlling the operation of the slave nodes. Each slave node carries out a specific function independent of the other slave nodes. Each node transmits and receives only the data it requires from the communication bus. The major advantages of this architecture are that it achieves the modularity and extensibility of the distributed architecture while retaining autonomous control of the individual nodes in response to their status and/or the requirements of the system at that time. The major drawback of this approach is the large component count and the large amount of circuit board space required.

Architecture Selection

The Master-Slave architecture was chosen from the above architectures for the following two reasons:

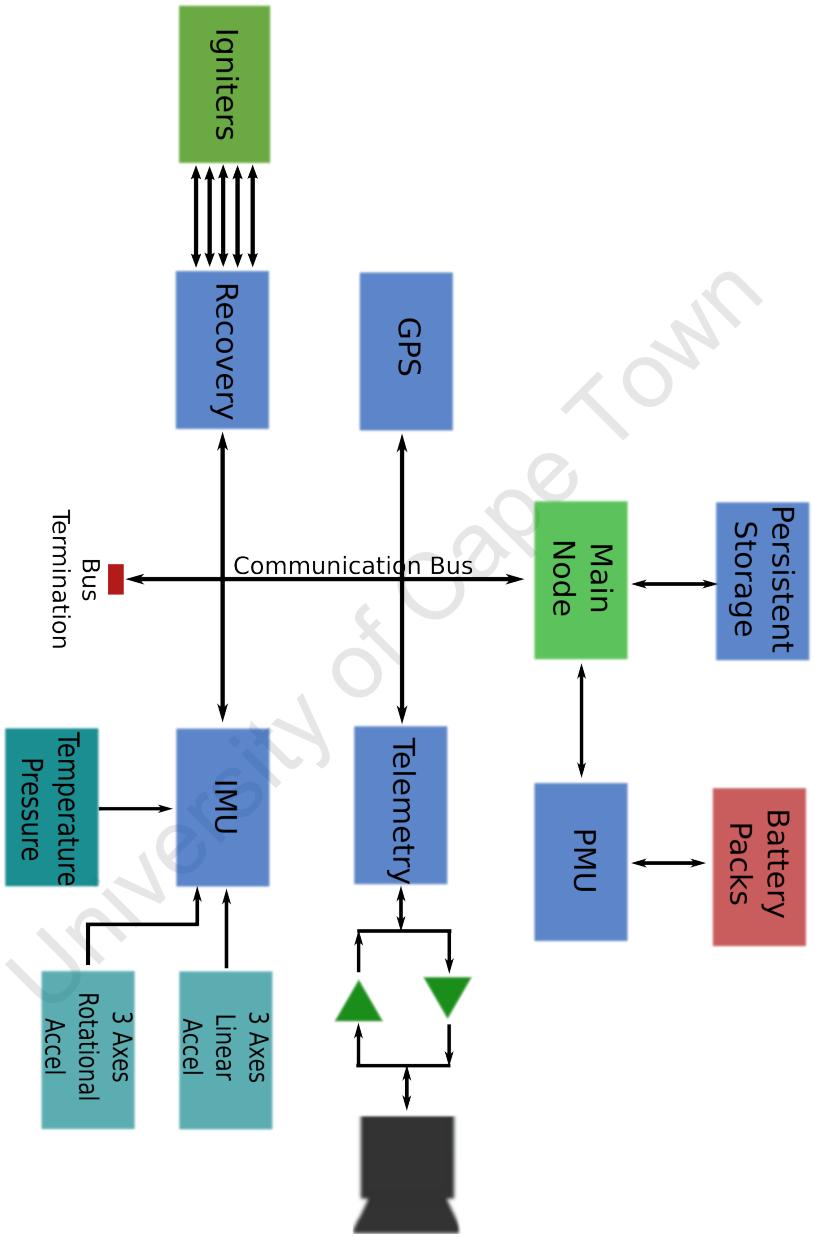

Figure 4.4: Master-slave architecture.

- The ability of the Master-Slave architecture to achieve the modularity and extensibility requirements of SAAO, as set out in Requirements R12 and R20.
- The ability of the Master-Slave architecture to control autonomously the functions of the flight computer by monitoring their states. This is linked to the requirements of function F6.

Communication Bus Review and Selection

The selection of the data bus is essential to fulfilling the requirements for data integrity highlighted in requirement R18. Further, the need for a standard well-known communication protocol is illustrated in requirements R12, R19 and function F6. That is, the communication protocol cannot be a rare obscure protocol if it is to be easy for users to extend the flight computer.

I2C

I2C is a bus communication protocol first introduced by Philips. It is a two-wire protocol including a clock line and a data line [11]. Each node on the bus is addressed with a 7-bit address, though larger addressing schemes exist. The 7-bit address space limits the number of nodes that may be added to the bus to 128. Nodes may be either a master or a slave on the bus. It is possible to have more than one master node on the bus, in which case bus arbitration is carried out by giving the master node addressing the slave with the lowest value address priority. The bus generally operates up to a maximum rate of 100kb/s, however variations exist that operate above 1Mb/s. The protocol does not include any data checking or resending of messages. It is up to the user to include this functionality on top of the protocol. However it should be noted that adding this functionality in firmware will increase processor overhead. Given that data integrity is a requirement as per requirement R18, this is an important point as this functionality must be added. This would then increase the power consumption of the flight computer because the extra processing overhead would be added to each node. This would hinder the fulfilment of requirement R13.3.

RS-485

RS-485 is a communication standard defined by EIA [16]. It uses a differential pair to transmit signals on the bus. The bus requires 3 lines, including the differential pair and a reference voltage. The fact that RS-485 uses a differential method of transmission means that it is far more noise resistant then I2C. Further, this noise immunity enables the bus to transmit at a much higher data rate; in this case, 35Mb/s if the nodes are within 12m. A maximum of 32 nodes may be added to the bus. The major drawback of this standard is that it does not implement any communication protocol, it is just a hardware standard. This means that all functionality must be implemented by the user in firmware. Therefore as with I2C this makes it more difficult to fulfil the data integrity power consumption requirements.

Controller Area Network

The controller area network (CAN) bus was initially design by Bosch for applications in the automotive industry. It was designed to enable microcontrollers to communicate with each other without the need for a host computer [17]. Communication is done over a differential bus giving CAN a high resistance to external noise as with RS-485. Each node is given an address of up to 16 bits. This means CAN is capable of supporting a very large number of nodes. Over short distances communication at up to 1Mb/s is possible. The CAN protocol implements a transfer layer, in hardware, that includes features such as error detection, message acknowledgement, message retransmission and bus arbitration. These features are well suited to fulfilling the data integrity and power consumption requirements highlighted above.

Communication Bus Selection

From the above communication buses the CAN bus has been selected as the communication bus most capable of fulfilling the requirements set out above. This is due to its use of differential communication, and the implementation of error correction and data retransmission in the transmission layer of the CAN protocol.

Microcontroller Review and Selection

Microcontrollers are available from a wide range of companies such as Atmel, Microchip Technology, Freescale, NXP and Texas Instruments. Each of these companies supports the communication bus standard selected above as well as numerous communication standards that are anticipated to be required in the course of this project. These include communication standards such as Serial Peripheral Interface (SPI). However, the support of the University of Cape Town for Freescale Microcontrollers means that the writer is most familiar with Freescale devices. This, coupled with the ability of Freescale's devices to fulfil the needs of this project, has led to a Freescale device being selected as the node standard. The device chosen to power the nodes in the flight computer is the Coldfire V1 MCF51JM128 [18].

Pin Header

In order to facilitate modularity set out in requirements R12, R20 and R21, a common bus connection is required. What follows is the arrangement of the signal lines on the flight computer bus. These include power channels, communication lines and logic signals.

1. CAN HI
2. CAN LOW
3. CAN GND
4. RF GND
5. Open Logic
6. ARM EN
7. Open Logic
8. External Power
9. Power GND 1

10. Power GND 2
11. Power 1
12. Power 2
13. Power 3
14. Power 4
15. Power 5
16. Power 6

Each of the power channels in the above list provides 5V and supports a maximum current of 200mA.

Basic Node Building Blocks

The common bus connection described entails that certain common hardware blocks will be required on each of the nodes of the flight computer. This section lays out the common hardware blocks required for any node on the flight computer bus.

As shown in Figure 4.5, each node requires a minimum of a single microcontroller capable of operating at or under 5V and capable of using the Controller Area Network (CAN) protocol. Further, each node requires a CAN transceiver to convert the serial data stream from the microcontroller to the differential signals used on the CAN bus. Each node requires some method of power channel selection which may be as simple as a set of jumpers. Each node may make use of any of the logic signals provided on the pin header. Finally, each node has a JTAG socket for programming, and provides a USB jack for external power.

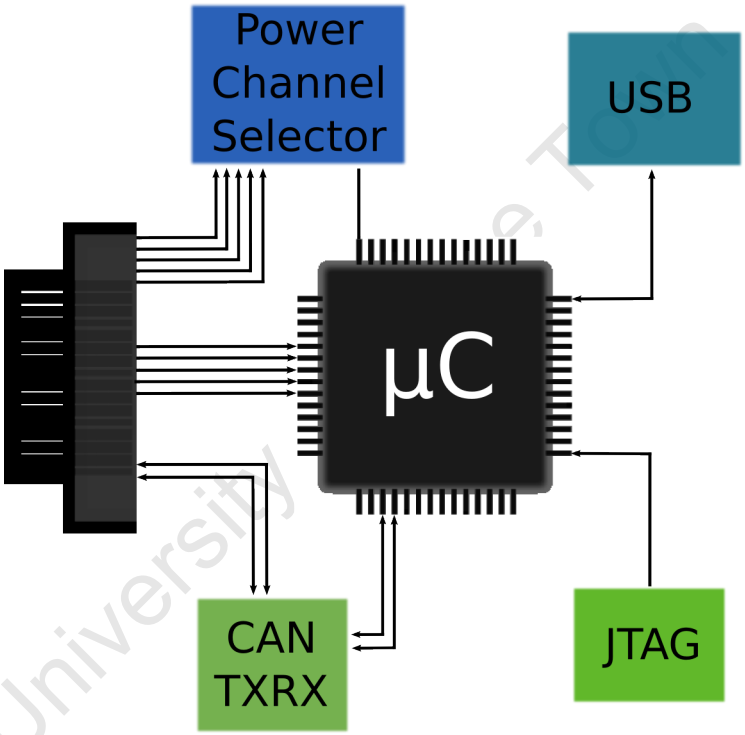

Figure 4.5: Basic node building blocks.

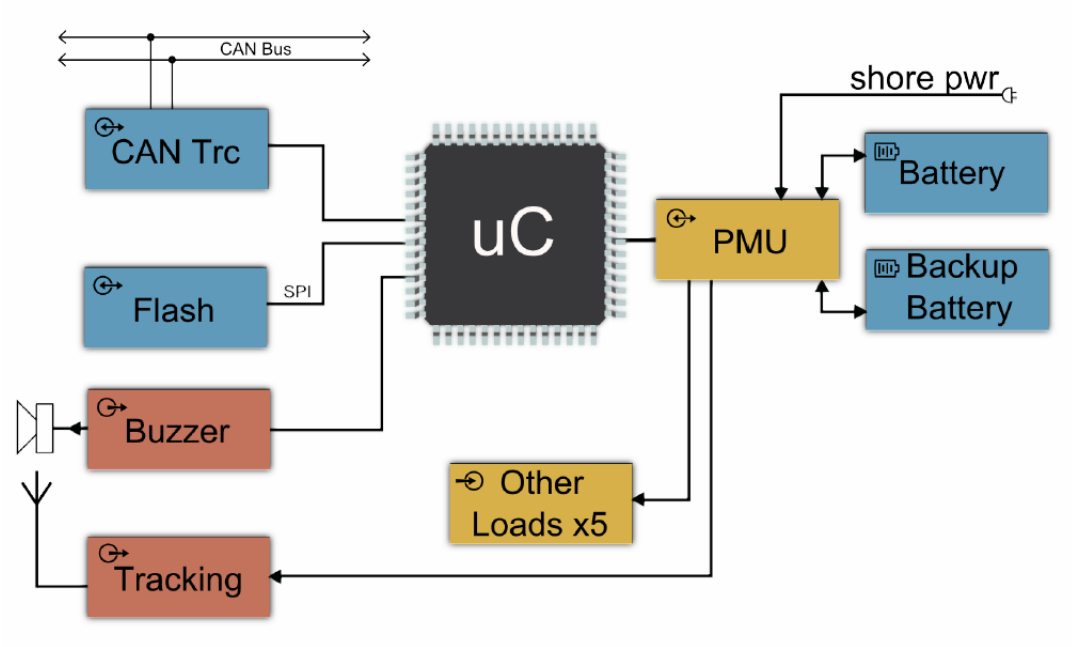

Figure 4.6: Block diagram of the main node [19].

4.3 Node Design

The Main Node

Figure 4.6 illustrates the features included on the Main Node.

The main node is the master node on the CAN Bus; it controls all the other nodes, including supplying power to each node and monitoring each node's status and power consumption. This node also controls the transmission of data to the ground by amalgamating data from the slave nodes into a data frame with a time stamp and then forwarding this frame to the telemetry node for transmission.

Power Management Unit

In order to fulfil requirements R7, R12 and function F6 the Power Management Unit (PMU) placed on this node has the following features:

Current and Voltage Measurement The PMU has the ability to measure the voltage level of both the main and the backup battery pack. Based on this information estimates may be made of the remaining operational time of the flight computer. Action may be taken by the PMU to save battery life in the case of a very low battery voltage. The PMU is also able to monitor the current consumption each of the power channels.

Channel Switching The PMU has the ability switch the power channels on and off. It does this through the use of a bank of high side switches. If the PMU detects that a slave node is drawing too much current it may switch it off through this function. It also provides a means of saving battery life when necessary.

Batteries The flight computer is designed to run with two Li-ion battery packs. Each battery pack must be capable of providing 5V for 100 minutes. Given that there are six power channels, and each power channel is limited to a maximum current of 200mA, this requirement necessitates a battery with a capacity of approximately 2000mAh. The choice of this type of battery is based on its specific energy. Li-ion batteries have a very high specific energy. This means that they can store a lot more electrical energy per unit mass than, for example, a lead acid battery. This property makes Li-ion batteries well suited for use in the SAAO flight computer given the weight constraint of the SAAO user requirements.

The batteries are charged through two independent Li-ion battery chargers on the PMU. Charging is enabled when the flight computer is connected to external power. The charging status of the batteries is passed from the PMU to the microcontroller so that the user can see when the batteries are charged.

Power Regulation The 5V required by the flight computer is provided by a low drop-out 5V regulator on the PMU. It was decided to use a linear regulator despite its inefficiencies due to the fact that it was unclear how much effect the switching noise from a switch mode power supply would have. However, in future revisions of the flight computer the use of a switch mode power supply should be pursued. Any of the

nodes that require 3.3V regulate the 5V signal down to 3.3V independently through the use of low drop-out regulators.

Data Storage

The main node is the centre for data storage on the flight computer, to this end a micro-SD card has been placed on this node. The reason for the use of SD cards as persistent storage was the ability to increase easily the size of the storage. Further, it is easy to remove the SD card to retrieve the persistent data should the rest of the flight computer be damaged in some way. The limitation of this method of storage is that it is not as mechanically robust as an IC soldered onto the circuit board. However steps may be taken to secure the micro SD card as necessary. For example, in the case of the SAAO flight computer provision has been made to bolt the SD card into place.

Status Indication

A buzzer has been placed on the main node to fulfil the requirements of function F21. It has been placed on the main node because the main node will manage all the other nodes. This node will therefore be the place to which status information is sent. The main node will make the status of the flight computer known to the user audibly through the use of this buzzer.

Extra Power Connectors

Two power jumpers are provided to the users in case they wish to power external components, such as a tracking beacon.

Recovery Node

Figure 4.7 shows a diagram of the recovery node.

The following are the features found on the recovery node:

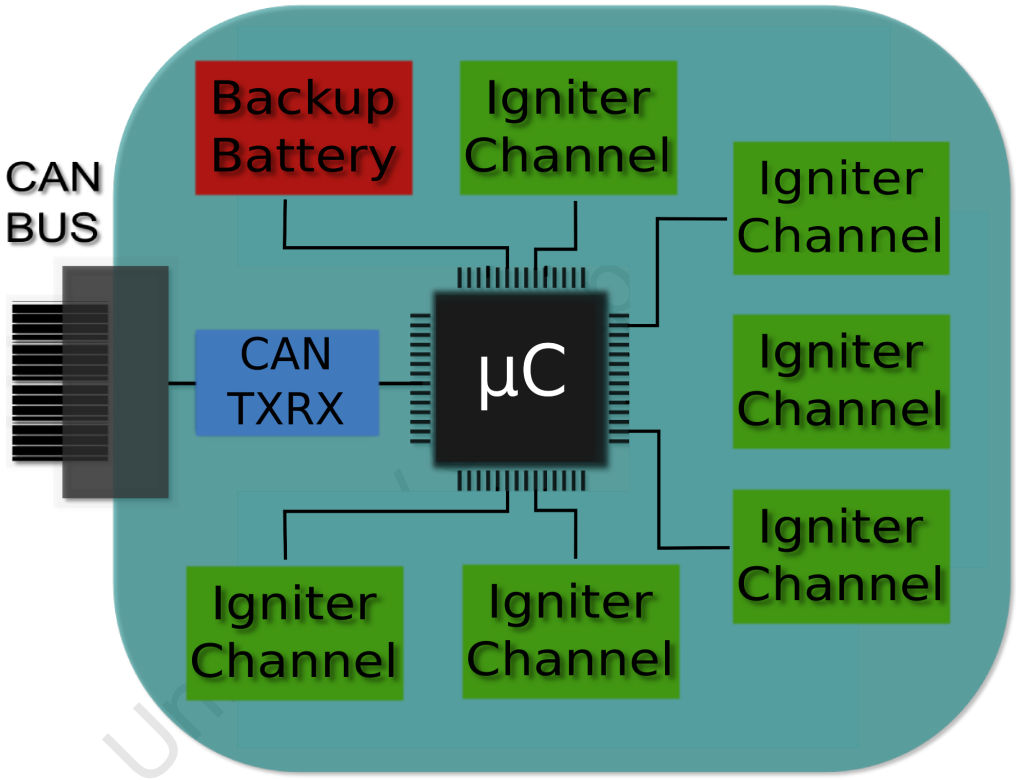

Figure 4.7: Block diagram of the recovery node.

Igniters

Igniter Channels The recovery node has six independent igniter channels, each capable of providing 3A for 0.5s. This is in fulfilment of requirement R9. Each channel may only be fired a single time, and only one channel may be fired at a time. The current is measured on each channel to monitor and limit the current consumption of that channel.

Igniter Interlocks The firing of the igniter channels is controlled by a number of interlocks. The following process must be followed in order to fire an igniter channel.

1. First the user must set the arm enable logic signal high.
2. Next the fire enable line on the recovery node should be set to a logic high.
3. Next the user needs to select an igniter channel and enable it with a logic high.
4. Finally the user needs to issue the fire command via a logic high on the fire line. Only with all the other interlocks at a logic high will this command work.

The reason for this complicated procedure is the safety of the users. These interlocks eliminate any accidental firings of igniter channels.

Independent Power Supply

Due to the fact that the operation of the recovery system is crucial to the successful flight of the rocket, the recovery node has been given its own independent backup power supply. In the event of a power failure from the main node the recovery node will switch to the backup power supply. The power supply to the recovery node includes current and voltage sensing of the supply and the ability to charge the battery from an external power supply. This design is to fulfil requirements R14 and R16.

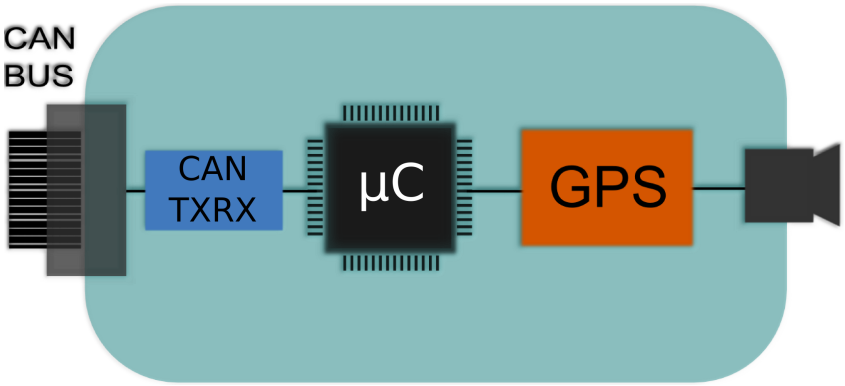

Figure 4.8: Block diagram of the GPS node.

Status Indication and External User Control

In order to provide the user with a clear indication of which igniter channels are armed and to provide the user with the ability to control mechanically which igniters can fire, the user has been provided with a set of jumpers and indicator LEDs. This is primarily for use as a safety measure when working on the flight computer on the ground.

GPS Node

Figure 4.8 shows a diagram of the GPS node.

The following are the features found on the GPS node:

COTs GPS

The GPS node makes use of a single COTs GPS receiver. This does not meet the performance requirements laid out in requirement R13. However, it was decided that the design of a GPS receiver that would operate within the constraints of SAAO was outside of the scope of this dissertation. This was due to the cost and the time it would take to design a GPS receiver capable of operating within the SAAO constraints during the boost phase of the rocket flight.

Status Indication

The GPS node provides a visual indication of the number of GPS satellites to which the node is currently locked. This is primarily for testing and integration of the flight computer with the rocket body to ensure that the configuration of the user's antennas allows for GPS lock.

Telemetry Node

Figure 4.9 shows a diagram of the telemetry node.

Frequency of Operation

ISM Band Selection Although there is now a telemetry band in the C band, at the time of the design of the telemetry module this band was not available. Further, there are no easily available, off-the-shelf integrated circuits that operate at this frequency. This makes designing for this frequency much more complex and so outside the scope of this dissertation.

The main reason for only considering the ISM bands is the availability of off-the-shelf integrated circuits that operate at these frequencies.

There are three main considerations in the analysis of the ISM bands. Firstly, the free space path loss of the frequency which effects how much power is required to transmit over long distances. Secondly, the available data rates at the frequencies under consideration. The telemetry module requires a data rate high enough to deal with the expected data from the flight computer with margin to handle re-transmissions for error correction. Finally, the transmission protocols available in the bands are considered. It is preferable that error checking and correction is implemented in hardware. This is because if this is implemented in firmware on the telemetry node processor it will increase the processor overhead on telemetry node processor. This in turn increases the power consumption of the node.

The two bands considered were the 433MHz and 2.4GHz ISM bands. No higher bands

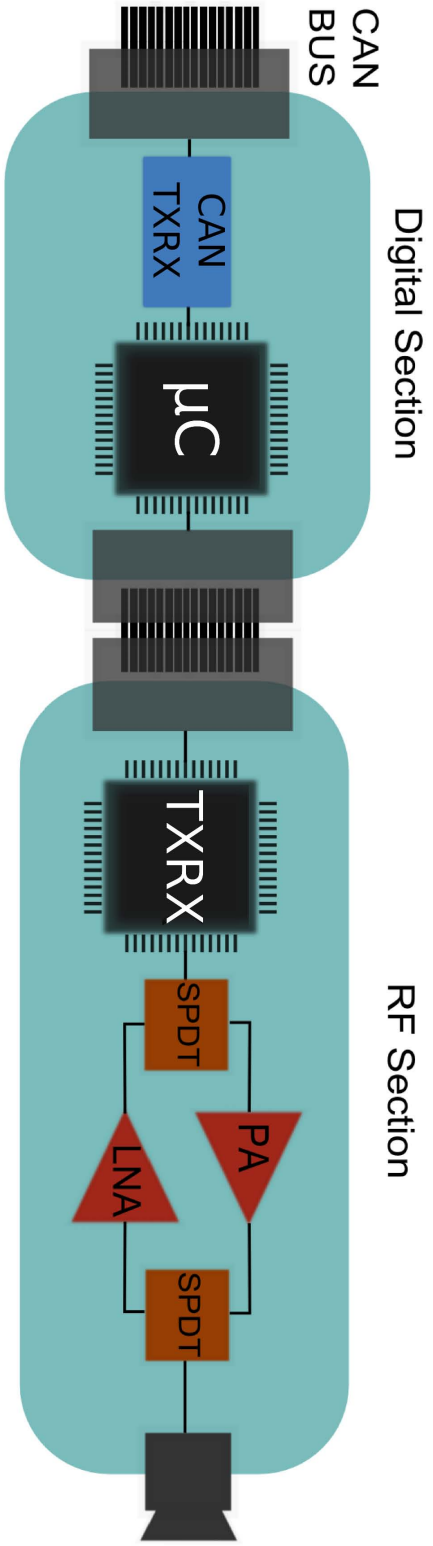

Figure 4.9: Block diagram of the telemetry node.

were considered due to the complexity of designing for higher frequencies.

433MHz The 433MHz ISM band is probably the most commonly used band for amateur rocketry. The availability of simple transceivers and the relative simplicity of designing for the 433MHz band is a great advantage here. A number of companies, including Nordic Semiconductors, Radiometrix and Texas Instruments, produce devices that operate at this frequency. The devices tend to be simple with little or no transmission protocol. The data rates tend to be in the range of 10-100kbps. The main advantage of this band is the low free space path loss at this frequency.

2.4GHz The 2.4GHz band provides a range of powerful transceivers capable of data rates up to 2Mbps. Transceivers are available from companies such as Texas Instruments and Nordic Semiconductors. The devices in this band tend to have more complicated transmission protocols, including cyclic redundancy checking and error induced retransmission. The main drawbacks of this band are the power limitations of the ISM band and the high free space path loss relative to the 433MHz band.

Band Selection The band selected for the telemetry node is the 2.4GHz band. This is due to the availability of transceivers that implement error correction and message retransmission in their transmission layer. This selection is in aid of meeting requirements R13.3 and R18. Further, the transceivers in this band are capable of data rates up to 2Mbps. These data rates are capable of managing both the data load from the flight computer and added load imposed by the retransmission of messages.

Architecture

As shown in Figure 4.9, the telemetry node operates in a half-duplex mode through a switching architecture [20]. The reason for the switching architecture is to allow the addition of a power amplifier output stage and a low noise amplifier input stage. The switches are controlled by a logic signal from the transceiver which also controls their power supply through a high-side switching configuration.

RF Component	S11 (dB)	S21 (dB)	Isolation (dB)
PA	-25.763	21.189	
LNA	-22.248	22.409	
SPDT	-19.7	-0.35	-26

Table 4.1: Simulated RF output stage S parameters.

The RF section of the telemetry node is separated from the digital section of the telemetry node. This is to ensure a minimum amount of noise transfer from the digital circuitry to the RF circuitry.

Devices from both Texas Instruments and Nordic Semiconductors were considered. Both companies provide transceivers with data rates up to 2Mbps, and transmission protocols capable of error correction and packet retransmission. However, due to the ease of availability of the Nordic Semiconductors devices in South Africa it was decided that they should be used. The device chosen for the Telemetry module is the nRF24L01+ [21].

Performance

The amplifiers used in the output and input stage of the telemetry node were simulated using Agilent's Gensys. The power amplifier selected for the flight computer is the Avago MGA-83563 [22]. It is a 3.3V 23dB power amplifier suitable for the modulation type used in the telemetry node. The low-noise amplifier is an Avago MGA-86563 5V 20dB low-noise amplifier [23]. The following table shows the simulated performance of the amplifier configuration. To see the simulated S11, S21, curves refer to the appendices. The expected results for the SPDT are taken from the data sheet [24].

Link Budget

Table 4.1 shows the expected performance of the RF output stage of the telemetry node. Table 4.2 shows the parameters required to calculate a simple link budget. The antenna on the flight computer side is assumed to be an omni-directional antenna with unity gain, and the antenna at the ground-station is assumed to be a yagi antenna with a

Link Budget Component	
Transceiver Power	0 dBm
Simulated PA Gain	21.189 dB
Receiver Sensitivity at 1Mbps	-85 dBm
Antenna Gain	15 dBi
Simulated LNA Gain	22.409 dB
Simulated SPDT Power Drop	1.4 dB

Table 4.2: Link Budget Parameters

gain of 15dBi. The value for simulated SPDT power drop is the expected drop across four switches given the architecture shown in Figure 4.9. The maximum possible range with these parameters in rocket flight would be approximately 10km.

Inertial Measurement Unit

Figure 4.10 shows a diagram of the inertial measurement unit.

The following are the features found on the inertial measurement unit.

Linear Acceleration Measurement The inertial measurement unit provides measurement of acceleration in the X, Y and Z axes. In the Z-axis two accelerometers are provided, one with a very large range of $\pm$200g and the other with a smaller range of $\pm$20g. This is due to the fact that the accelerometer with the smaller range has a much higher measurement resolution over its range. This is useful for the flight phases where there is no motor burn as it allows a more accurate measurement of smaller accelerations.

Rotational Acceleration Measurement Rotational acceleration measurement is provided for three rotation axes, that is, pitch, yaw and roll. The SAAO has specified that their rockets will not be spin stabilised. Therefore rotational acceleration measurement up to 300°/s has been provided in the pitch and yaw axes and up to 2000°/s in the roll axes.

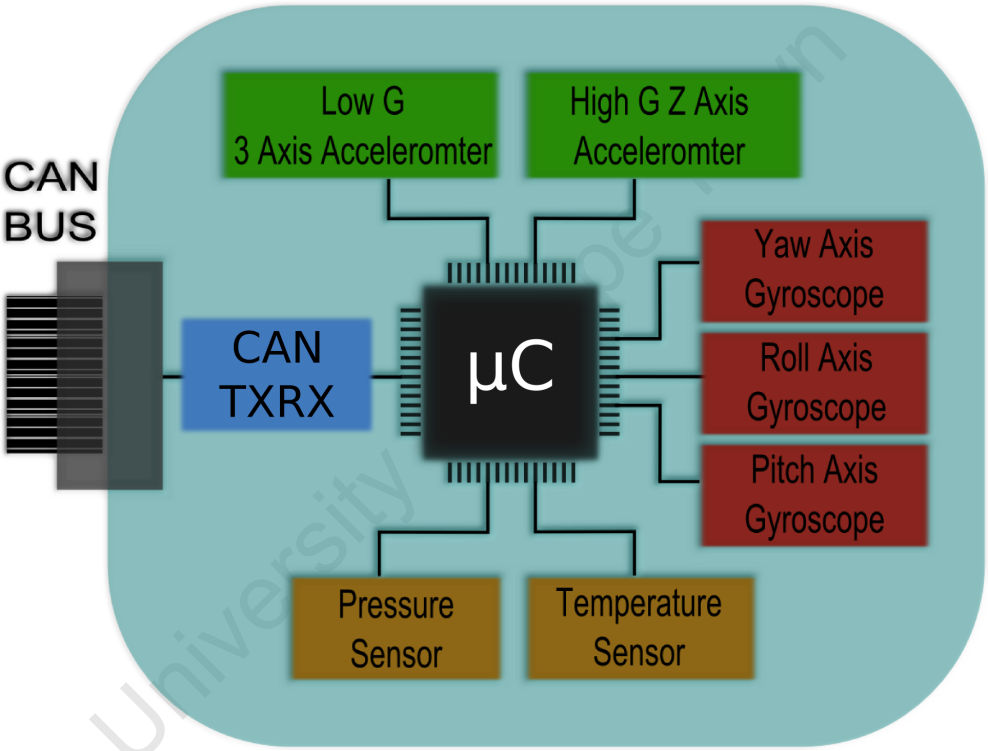

Figure 4.10: Block diagram of the inertial measurement node.

Pressure Measurement The inertial measurement unit provides a single absolute pressure transducer. The sensor is able to measure the pressure down to 0 ± 0.8kPa . The reason for selecting a sensor with this range is to ensure that the pressure transducer does not limit the maximum operational altitude of the inertial measurement unit.

Temperature Measurement Temperature measurement is provided on the inertial measurement unit. It is capable of a range of -40 to 105 $^{\circ}C$. The temperature sensor has been placed on the inertial measurement unit in order to provide a means of temperature calibrating the inertial measurement unit. It is the objective of the temperature sensor to measure the ambient temperature in which the inertial measurement unit operates.

Data Processing

The inertial measurement unit uses a linear constant gain Kalman filter on the pressure and acceleration data in order to calculate altitude. A second filter processes the rotational acceleration together with the linear acceleration data in order to determine the attitude of the rocket. This is done to provide a more accurate altitude measurement by converting the acceleration of the rocket into the Earth's reference frame.

GPS measurements are provided to the inertial measurement unit to reduce position error caused by position calculations based on inertial measurements. The data are to be fused using the Kalman filter.

Outputs

The inertial measurement unit provides outputs on the serial bus as well as providing a number of timer channel outputs. These could be used to control servo motors gimballing a rocket motor, or adjusting fins. It is, however, beyond the scope of this dissertation to implement their application.

Ground Station

Figure 4.11 shows a diagram of the ground station.

The following are the features included on the ground station.

GPS

In order to provide differential GPS measurements a GPS receiver is required at the ground station. The user will input the known position of the fixed ground station into the flight computer. The known position will then be differenced with the measured location of the ground station and the difference will be forwarded to the flight computer. This approach assumes that the flight computer has lock with the same GPS satellites as the ground station. In this way the GPS offset can be eliminated and a much more accurate position measurement may be achieved.

Telemetry

The ground station has a telemetry node attached to it to facilitate communication with the rocket section of the flight computer.

Data Processing

The ground station forwards data received via the telemetry link to a computer. The computer handles all the data processing required in real time.

Graphical User Interface

The graphical user interface displays all the processed data to the user. This is done in real time to enable the user to respond to in flight events or to override manually the flight computer's behaviour in-flight. The graphical user interface displays all the flight data transmitted to the ground station by the flight computer.

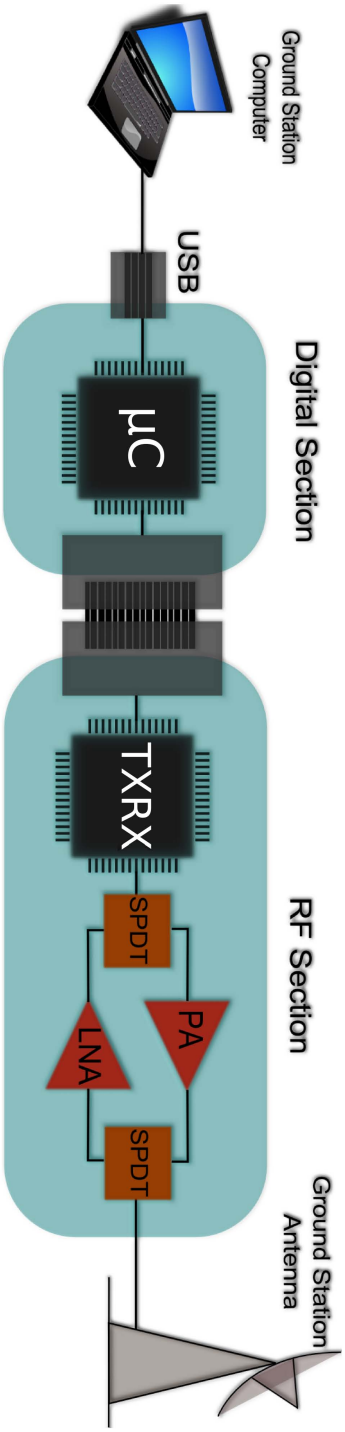

Figure 4.11: Block diagram of the ground station.

4.4 Firmware Design

The firmware for the flight computer uses a time-triggered architecture [9]. The reason for the use of this architecture over a pre-emptive architecture is the increased reliability and decreased system overhead. A more thorough treatment of this is given in chapter 3.2.

The flight computer uses a shared clock scheduler in order to synchronise the clocks of each of the nodes [9][10]. This provides a way to control communication on the CAN bus in a time-triggered manner. The main node broadcasts a timing CAN message on the CAN network. Each of the nodes on the network has a time triggered scheduler that is triggered by the main node timing message. Thus the clocks of each of the nodes are synchronised to the main node. Assuming a 1Mbps bus baud rate and an 8 byte CAN timing message there will be a delay between the main node trigger and the slave node triggers of about 150µs. In order to reduce jitter on the slave nodes the timing message sent by the main node is always the same. This prevents the bit stuffing that the CAN protocol utilises from producing a variable length timing message [10]. The main node is therefore not able to transmit data with the timing message because this would affect the number of stuffed bits in the timing message.

The slave nodes broadcast their data on the CAN network in a time-controlled manner. This entails a separate time slot for transmission on the CAN bus for each node. Figure 4.12 shows a diagram of the transmission cycle of the flight computer. In the example cycle shown all the nodes use their assigned message slot. This, however, is not always the case.

The first CAN message is offset by 150µs because of the main node timing message that precedes it. Each message on the CAN bus is given a 200µs slot. The transmission cycle has a duration of 2ms, however, not every node transmits in every cycle. Table 4.3 shows the period of each node message. The recovery node only needs to transmit a status update to the main node and this only requires a very low update rate. The telemetry node only requires bus time when it has commands from the ground station or when it has to update the main node on its status. Both of these functions are put together into a single CAN message at a low update rate. The GPS node only receives

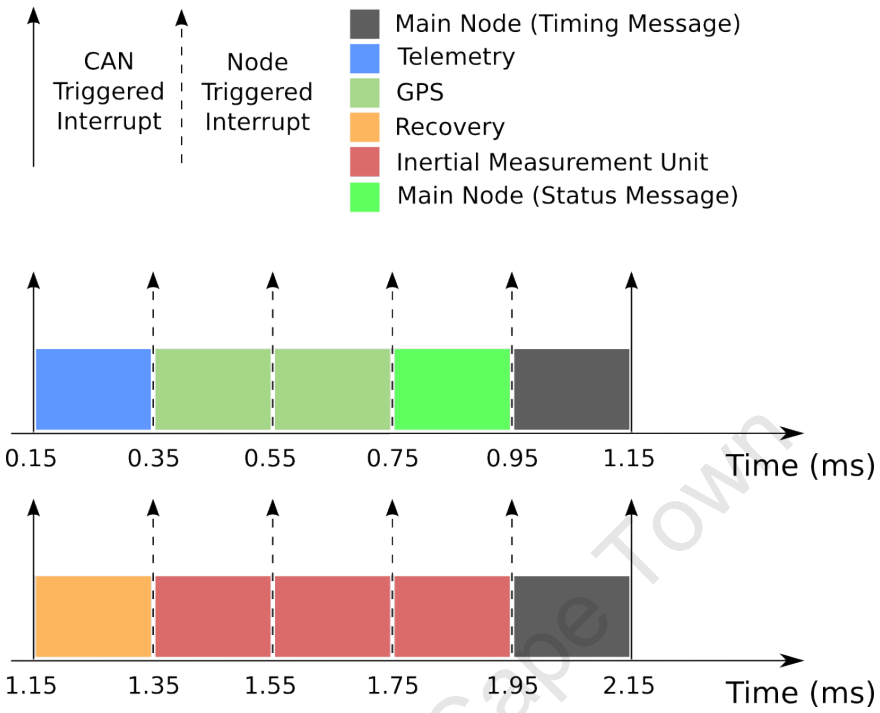

Figure 4.12: Block diagram of the CAN bus transmission cycle.

position updates at 1Hz, therefore it only requires bus time once every second. The status update of the GPS node is put together with the position data message. Finally, the inertial measurement node has an update rate of 500Hz. This is to provide high resolution position, velocity and acceleration data which in turn enables a high time resolution for event detection. Inertial measurement unit status updates and data are broadcast onto the bus every transmission cycle. It should be noted that the transmission layout detailed above leaves message space for future expansion of the flight computer.

Figure 4.13 shows the scheduling of tasks on the inertial measurement node. It has been chosen as an illustration because the period of its tasks are not widely distributed in time. The main node scheduler runs at 1ms tick intervals. Each tick triggers a CAN timing message to trigger all the slave nodes on the bus. Therefore the inertial measurement unit is triggered every 1ms. The schedule has a total period of 2ms. Each of the functions shown in the Figure 4.13 has a frequency of 500Hz, meaning every function is called every cycle through the schedule. The Master node status

Node Message	Frequency
Telemetry	1Hz
Recovery	1Hz
GPS	1Hz
Inertial Measurement Unit	500Hz
Master (Status)	1Hz

Table 4.3: This table shows the period of the node CAN messages.

message is transmitted using a different message identification number so that it does not trigger the node timers.

Inertial Measurement Unit Algorithm

In order to detect events in the flight of the rocket accurately the information from the various sensors on the inertial measurement must be combined and filtered. Figure 4.14 shows a block diagram of the inertial measurement unit algorithm.

First the sampled gyroscope and accelerometer data are combined to determine the attitude of the rocket with respect to the Earth's reference frame. This may be done with either a Kalman filter or a Complimentary filter. The attitude data is then used to convert the measured acceleration from the rocket's reference frame to the Earth's reference frame. The acceleration values are then combined with the altitude values derived from the pressure measurement [25] with a linear Kalman filter [26]. The filter is made linear here by ignoring the effects of drag in the dynamic model of motion of the rocket. This makes the filter less accurate, however, the advantage of this approach is the simplicity of implementation of the filter. Due to the time constraints of this project, simplicity is preferable.

The main purpose of the filter here is to determine the altitude of the rocket and thus apogee, however, filtered acceleration and velocity data is also available through the filter. This filtered data is passed to an event detection block for the detection of flight events such as launch, motor-burnout and apogee.

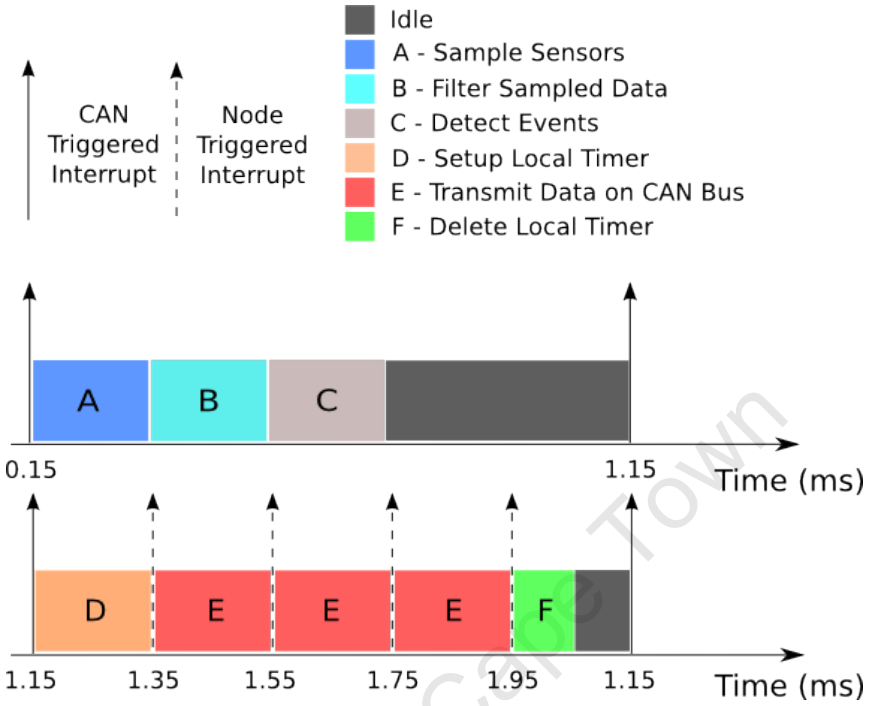

Figure 4.13: Block diagram of the inertial measurement unit task schedule.

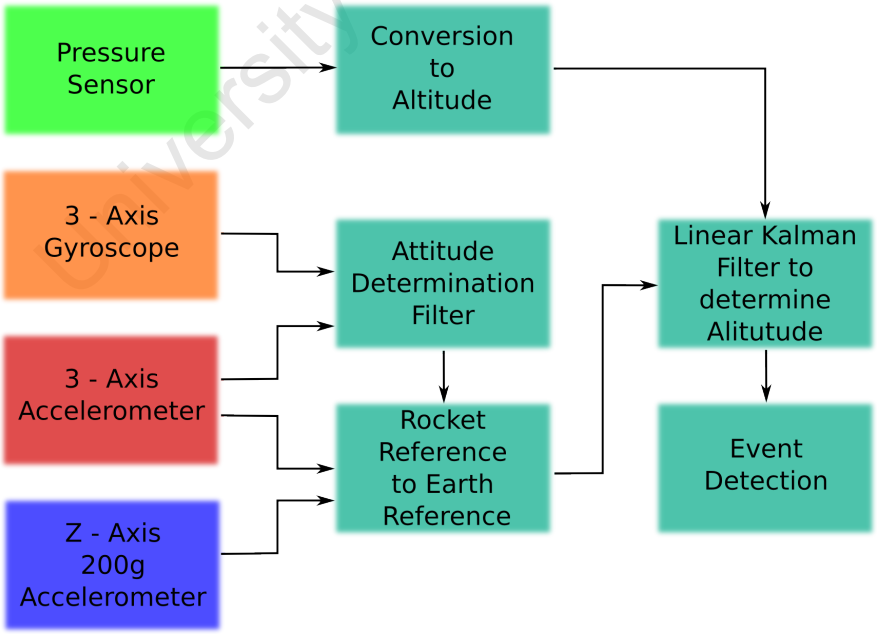

Figure 4.14: Block diagram of the inertial measurement event detection algorithm.

4.5 Conclusion

In summary of this chapter we have presented the design of the SAAO flight computer. The design has covered both the hardware architecture, the firmware architecture and the design of the algorithms required to process the on-board data. The design work done in this chapter has been based on the requirements and functional analysis given in chapter 3. In the next chapter we will look at the implementation and testing of the flight computer.

Chapter 5

Implementation and Testing

5.1 Introduction

The following chapter details the process followed to manufacture and test the prototypes of the digital circuitry. This is followed be the details of the testing carried out to verify the correct operation of the flight computer as laid out in chapter 3. Finally an analysis of potential failure modes and their effects on the flight computer is given followed by the means that have been taken to avoid these failures and suggestions for future work to prevent failures.

5.2 Prototyping

The circuit boards were fabricated on double-sided 1.4mm FR4 substrate. The fabrication process used was a routing process with a bare copper or tinned finish. The circuit boards were then through-hole plated to allow for circuitry on both sides of the board. The reason further layers were not used is that the cost would have been too great. However, using more layers would have substantially lowered the circuit board space required. Therefore increasing the number of layers in the circuit boards is recommended for future versions of the flight computer.

Prototype 1

Figure 5.1 shows the first prototype. The photo shows the master node and the recovery node.

Only the master, recovery, GPS, telemetry nodes and the ground station were fabricated for the first prototype. The RF sections of the telemetry node and the ground station were implemented using development boards.

Prototype 2

Figure 5.2 shows a photo of the completed second prototype of the recovery node. Prototype 2 involved a complete redesign of the circuit board layouts to use the circuit board space more efficiently and to reduce the width of the circuit boards. This change had no bearing on the fundamental design of the flight computer architecture or the flight computer's nodes. Consideration was given to the orientation of the circuit boards for mounting. It was decided that in order to give users access to the ribbon cable connectors, the pin-headers would be moved from the long axis of the circuit board to the short axis. This change is shown in Figure 5.2.

Prototype 3

Figure 5.3 is a photo of prototype 3 being prepared for a test launch. The photo shows the master node, recovery node, GPS node and telemetry node. The IMU is left of the image and is orientated horizontally rather than vertically as is the case with the other nodes. This is because of the axes for which the MEMs sensors are configured to measure. Prototype 3 kept the layout of prototype 2 but fixed the bugs found in the package layouts. This prototype was the last prototype fabricated for this dissertation.

The significant changes included in this prototype were associated with fixing the pin orientation in the mosfet packages used on the recovery node and refining the layout changes made in prototype 2. Before the mosfet changes were integrated with the main flight computer a test igniter switching channel was built on veroboard to verify

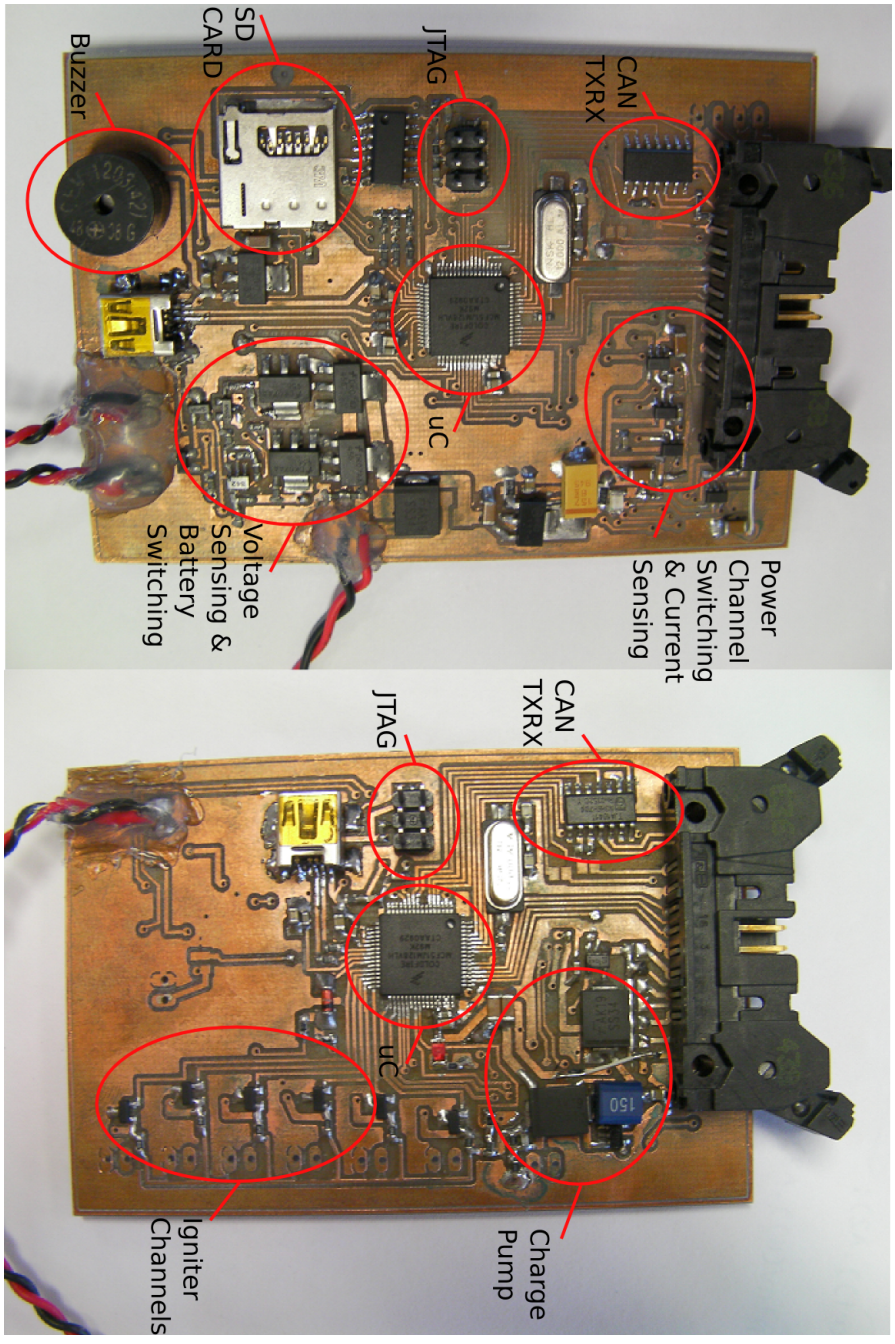

Figure 5.1: Prototype 1 [19]. The circuit board on the left is the master node and the circuit board on the right is the recovery node.

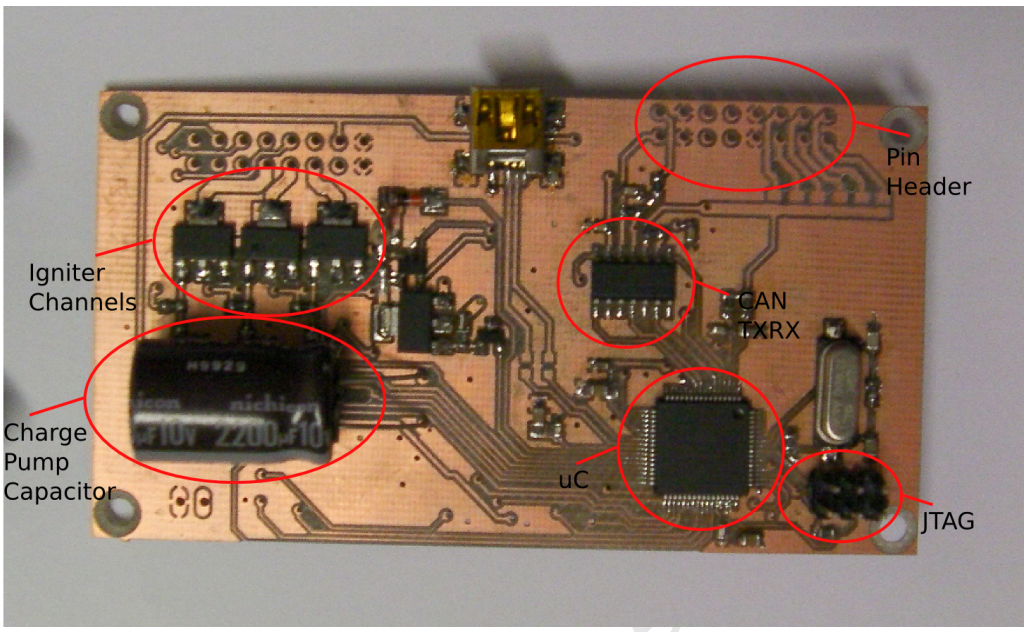

Figure 5.2: Prototype 2 circuit boards.

that the new set-up functioned correctly. Consideration was also given to the type of power connectors used. It was decided that SMA connectors would make a robust power connection when torqued to the correct tightness [27].

Prototyping of the RF Circuitry

The following section deals with the fabrication and testing of the RF circuitry associated with the telemetry link. This includes the power amplifier, low noise amplifier and RF switches used in the RF output stage.

Manufacturing Process The RF circuitry was fabricated using the same routing process as was used with the digital circuitry; however, the substrate used was 20mil Rogers 4003C. Testing and tuning was done through a process of testing the prototype boards on a network analyser and then adjusting the line lengths and passive components to improve the tuning.

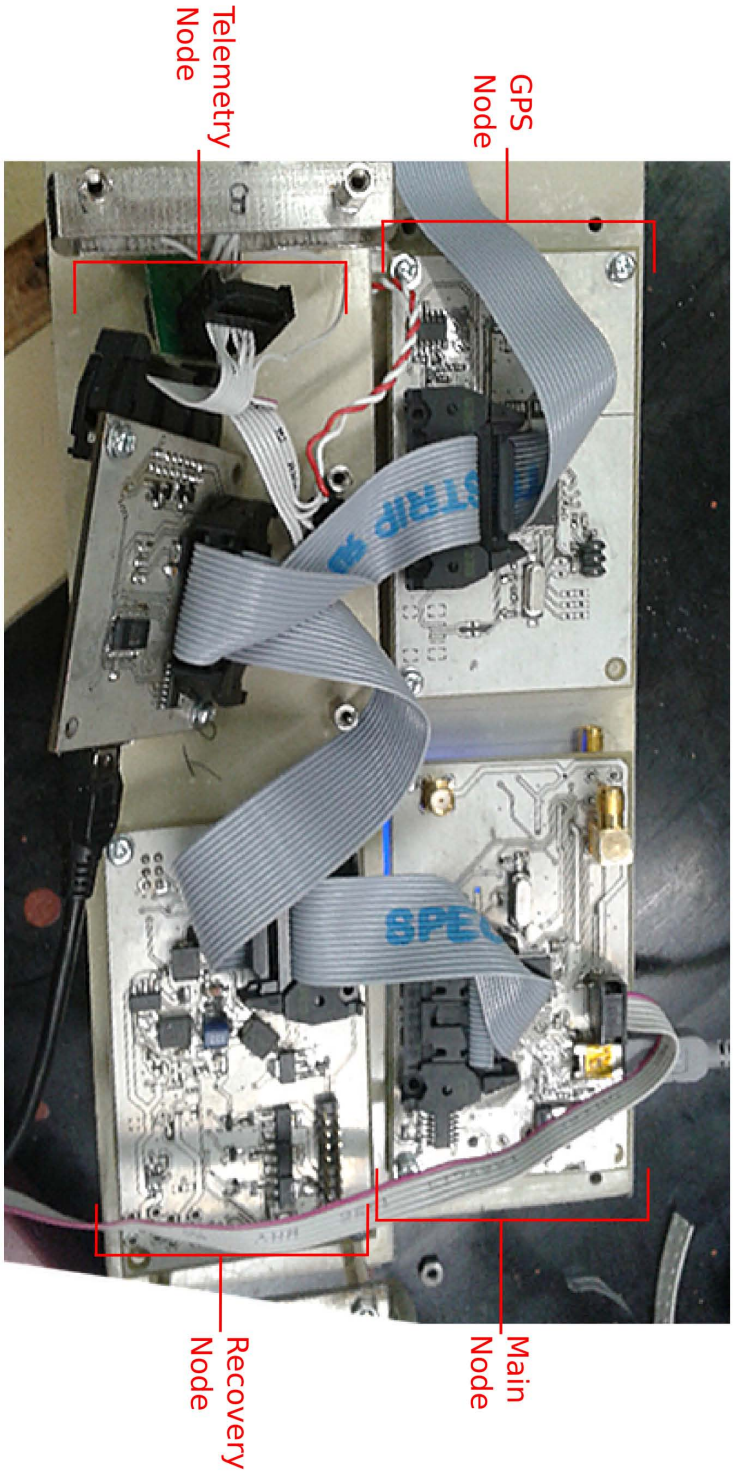

Figure 5.3: Photo of prototype 3 on the workbench.

Amplifier	S11 (dB)	S21 (dB)	Isolation (dB)
PA	-16.412	18.988	—
LNA	-14.610	27.509	—
SPDT	-16.268	-0.7101	-21.074

Table 5.1: Amalgamated RF output stage results.

Prototyping and Testing The first set of prototypes were fabricated on individual boards, that is each individual RF component was fabricated on its own board. This was done to ensure the tuning of each individual component was correct before amalgamating all the components onto a single RF output stage. A process of fabrication, testing and then re-fabrication was followed to achieve the best possible performance of the prototypes.

Results Once the performance of the individual prototypes was deemed to be acceptable, the individual RF prototypes were amalgamated into a single circuit board prototype. Table 5.1 shows the S parameters measured on the network analyser for the amalgamated RF output stage. For network analyser screen-shots of the performance of the individual RF components, refer to appendix B.

Even though the results shown in Table 5.1 are in many cases far short of the simulated ideal results, these results were deemed to be acceptable given that the manufacturing process used is not accurate enough to produce the boards within the required tolerances. For example, the milling process was unable to keep the line widths of the micro-strip lines consistent. Also because the placement and soldering of the passive components was done by hand there was variation in the length of the micro-strip between the pins of the RF IC and the passive components between the simulated design and the fabricated circuit.

The power amplifier return loss measured from the amalgamated prototype was -16.412dB versus -25.763dB simulated. However, -16.412dB still represents only 2.29% return from input power. The low noise amplifier return loss from the amalgamated prototype was measured to be -14.610dB versus -22.248dB simulated. This represents a 3.36% return from input power. The SPDT return loss was measured at -16.268dB versus -19.7dB simulated. This represents 2.36% return from input power.

These returns were all deemed to be acceptable given the limitations of the manufacturing process detailed above.

The gain of the power amplifier was measured at 18.988dB versus 21.189dB. This translates to an output signal power 79.21 times the power of the input signal. The low noise amplifier gain was measured at 27.509dB versus 22.409dB simulated. This measurement was taken for a -30dBm input signal. This is a noticeable improvement on the simulation. For the SPDT power drop it should be noted that the value shown in Table 5.1 is the drop across two SPDT switches. This is because in the amalgamated RF output stage, the input signal must pass through a SPDT before it passes through an amplifier. The output signal must then pass through another SPDT. The power drop measured across the two SPDTs together was -0.7101dB versus an expected -0.7dB. This represents an input signal power loss of approximately 7.54% and an ouput signal power loss of approximately 7.54% . This is a significant power loss, however, it is the expected loss when considering the simulation.

Graphical-User-Interface

Figure 5.4 shows a screen-shot of the ground-station graphical-user-interface (GUI). The GUI was written in C++ using version 4 of the Qt framework. The flight computer makes use of an FTDI serial to USB converter on its ground station, this simplifies interfacing with the computer. The GUI interfaces with the Linux POSIX driver in order to receive data via the serial port. The GUI includes the ability to log all the data received from the serial port to a file.

The left-hand column of tick boxes and numerical displays is dedicated to the control and status of the power management unit. The tick boxes provide control of each of the power management unit's channels. When a channel is activated its current is shown in the numerical display to the right of tick box. The voltages measured by the flight computer power management unit are displayed below the channel currents.

The right-hand column of tick boxes is for the control of the igniter channels. The tick boxes enable or disable each igniter channel. The software limits the user to enable only one igniter at a time. A further tick box is provided to enable the user to arm the

igniters and a fire button is provided to fire the enabled igniter channel manually. The lower right-hand numerical displays show the GPS position and inertial measurement output of altitude acceleration and velocity. System time is displayed in the bottom right-hand corner. This GUI meets the function requirements of F21 in that it provides a way for a user to control the flight computer.

In order to plot data in real-time and so better fulfil function F20, the GUI was extended to include a plotting function. This was written using the Qwt widget [28] for the Qt framework. Figure 5.5 shows a plot using the GUI plotting function.

5.3 Firmware Algorithms and Simulation Testing

The following section deals with the implementation of the firmware algorithms and the testing to verify their functionality. It also details all testing done in accordance with the testing procedures outlined in chapter 3. It should be noted that some tests were not possible because of the time constraints on the project.

Firmware Algorithm Implementation

Due to time constraints only the linear constant gain Kalman filter and the event detection algorithms shown in chapter 4.3 were implemented. These algorithms were first written as a PC application in the C language using floating-point arithmetic. This was done to expedite the development of the algorithms and to provide a platform on which to test the algorithms easily.

In order to test the functionality of these algorithms, real flight datasets were obtained through the RDAS mailing list. This data was then fed into the algorithms and the filtered outputs were plotted over the original data. Figure 5.6 shows a zoomed in section of a plot of the filtered altitude over the measured altitude at apogee. Figure 5.5 shows a zoomed in section of a plot of the filtered acceleration over the measured acceleration during the burn phase of the rocket. The green line in Figure 5.5 is the velocity of the rocket. The velocity is not directly measured, instead it is calculated

Figure 5.4: Screenshot of the test graphical user interface.

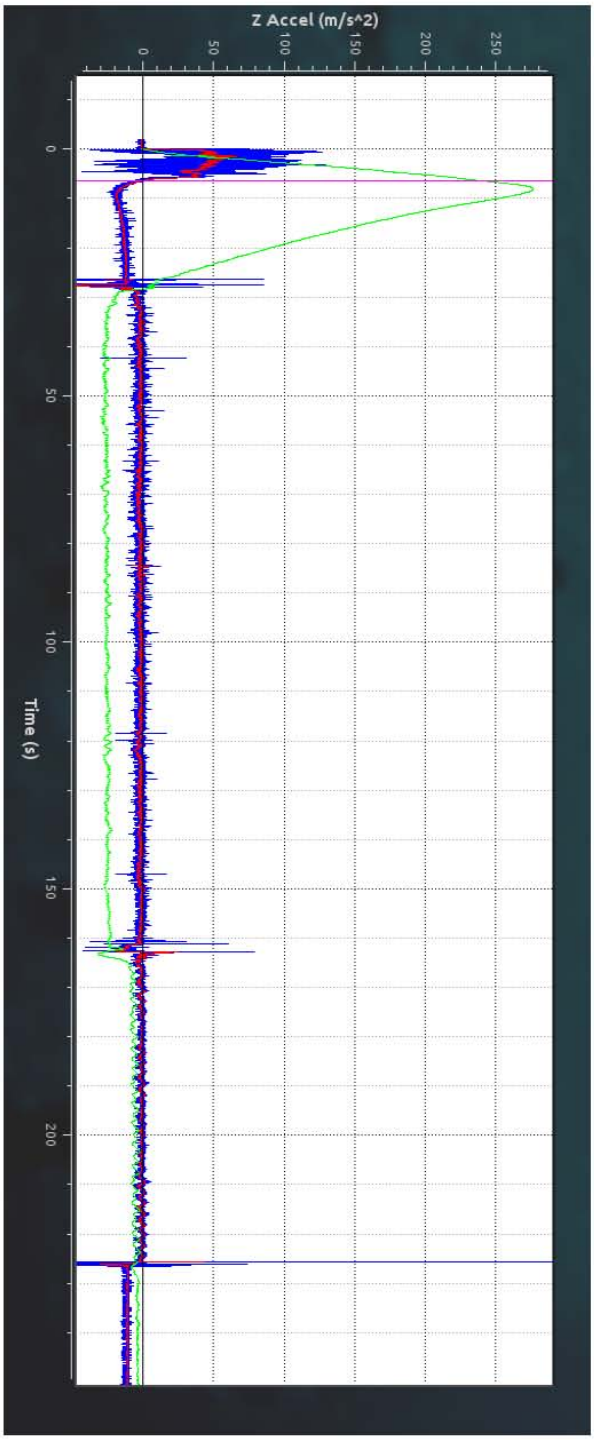

Figure 5.5: Plot of filtered acceleration (red), measured acceleration (blue) and derived velocity (green).

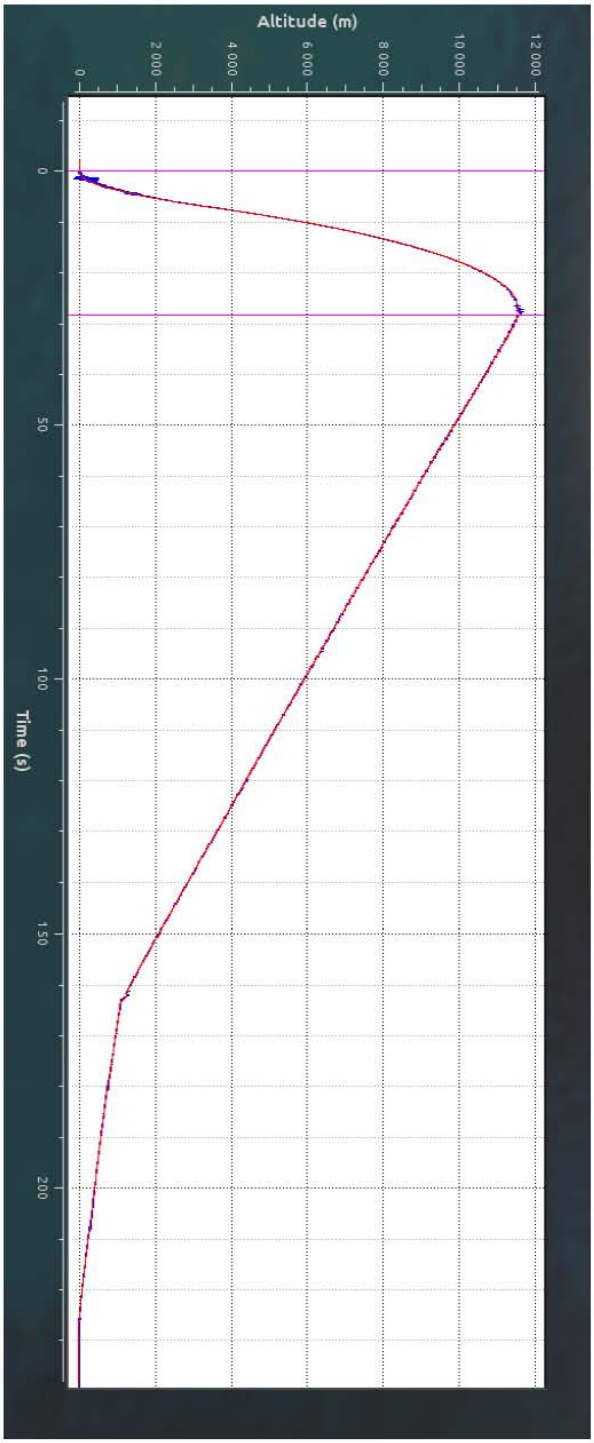

Figure 5.6: Plot of filtered altitude (red) and measured altitude (blue).

using the Kalman filter.

Once the filtering was deemed to function correctly, as shown in the Figures 5.6 and 5.5, the outputs of the filter were fed into an event detection algorithm. The purpose of the event detection algorithm was to detect all the events in requirement R2. The detection method for each of the functions is given below:

- R2.1 - Launch - Launch is detected by monitoring the filtered Z - axis acceleration for a sustained acceleration greater than 1g over 5 acceleration samples.
- R2.2 - Stage-Burnout - Stage-burnout is detected by monitoring the filtered Z - axis acceleration for a sign change in the acceleration measurement sustained over 5 acceleration samples.
- R2.5 - Apogee - Apogee is detected by monitoring the filtered altitude for a sustained decrease in altitude. This is achieved by taking 8 samples of filtered acceleration and averaging them over the time the samples were taken. If more than three consecutive sets of samples show a decrease in altitude then apogee is detected.
- R2.6 - Landing - Landing may only be detected after apogee is detected. It is detected by averaging the filtered altitude over sets of 5 seconds. If there is no detected change in altitude, then landing is detected.

No flight data was available for multi-stage rockets for use to refine detection algorithms for the stage-separation and stage-ignition event detection requirements.

In order to test the detection of these events the output of the detection algorithm was plotted over the filtered and unfiltered data. This may be seen in Figure 5.9, the vertical magenta line at time 6.287s represents the detection of apogee. The detection of launch may be seen in Figure 5.8 at time 0.0050s by the magenta vertical line. The detection of stage burnout may be seen in 5.7, the magenta vertical line at time 28.318s represents stage-burnout. This simulation process was repeated for fifteen different data sets and in each case the algorithms performed as expected.

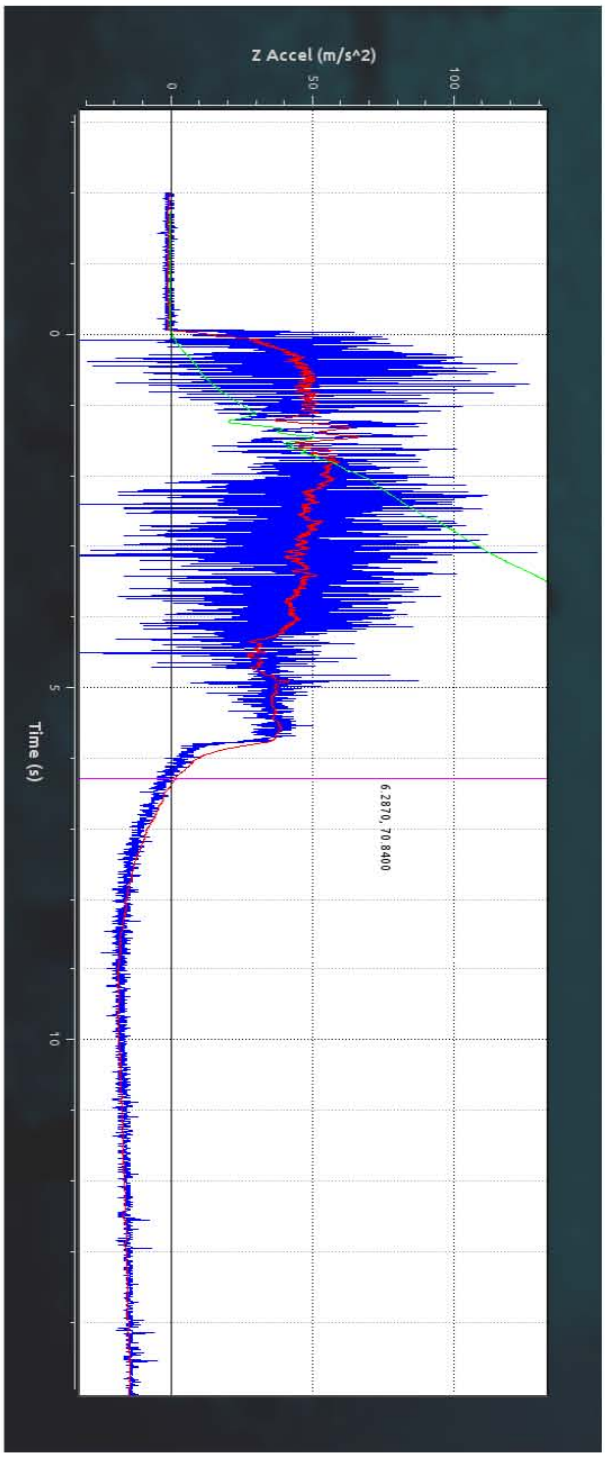

Figure 5.7: Plot of measured acceleration (blue) and filtered acceleration (red) of the burn phase of a rocket with burnout detection shown at 6.287s by the magenta vertical line.

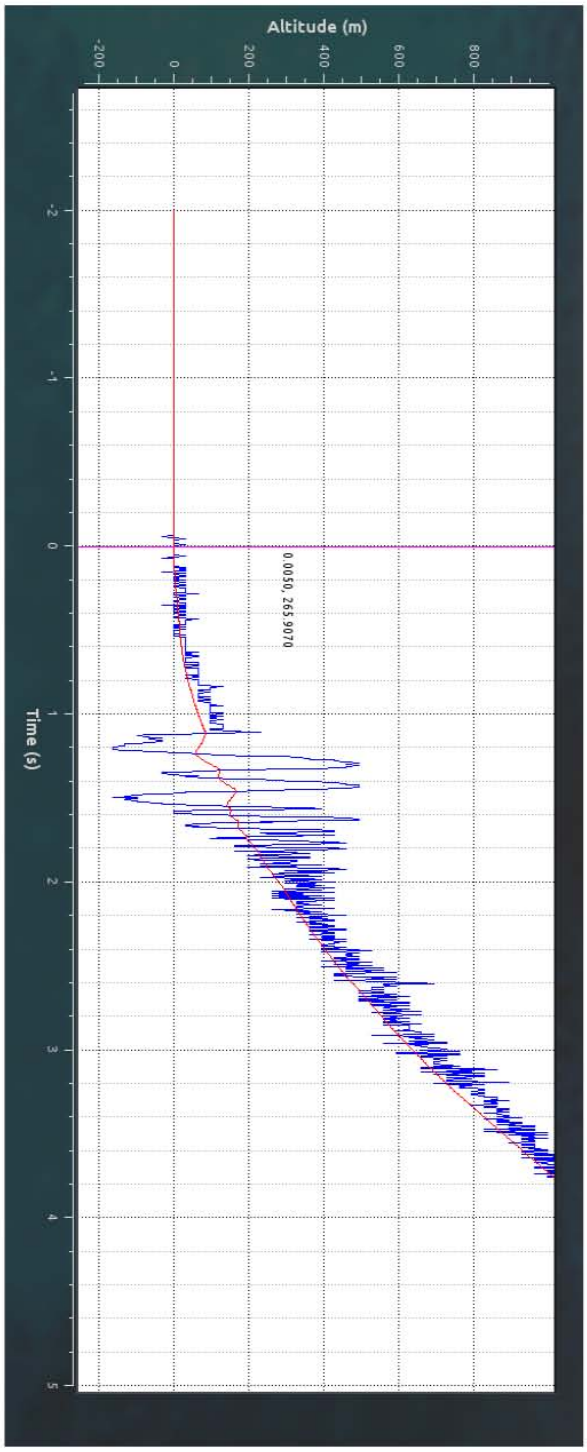

Figure 5.8: Plot of measured altitude (blue) and filtered altitude (red) of the launch phase of a rocket with launch detection shown at 0.005s by the magenta vertical line.

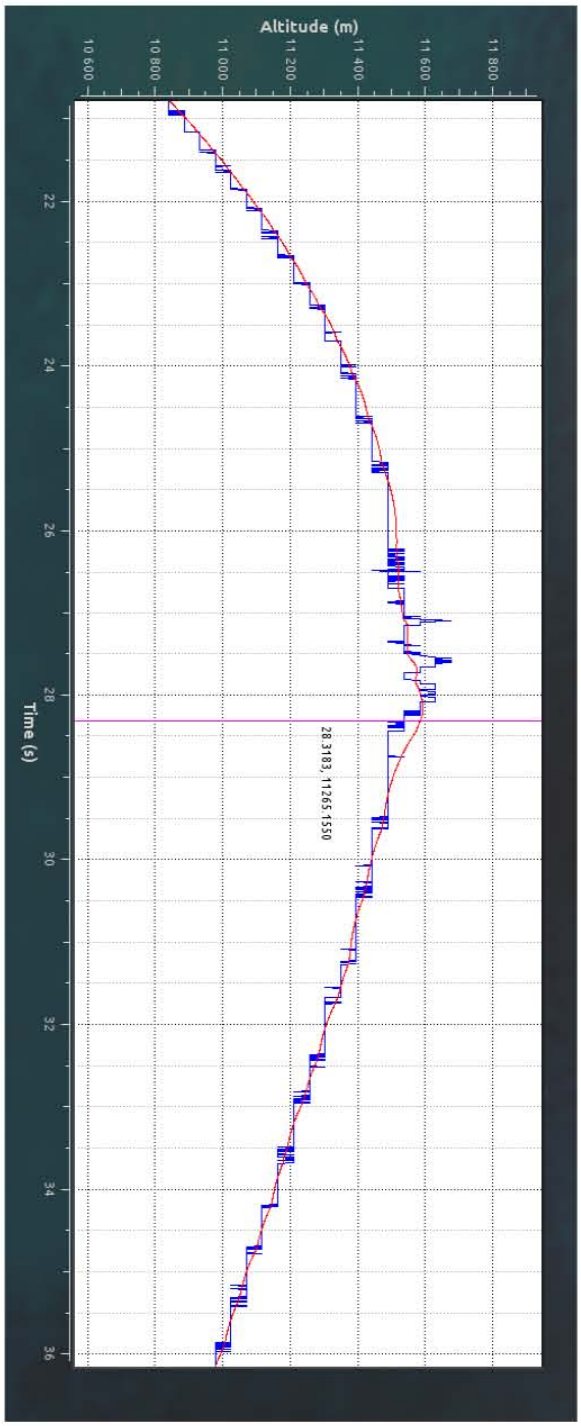

Figure 5.9: Plot of measured altitude (blue) and filtered altitude (red) of the apogee phase of a rocket with apogee detection shown at 28.318s by the magenta vertical line.

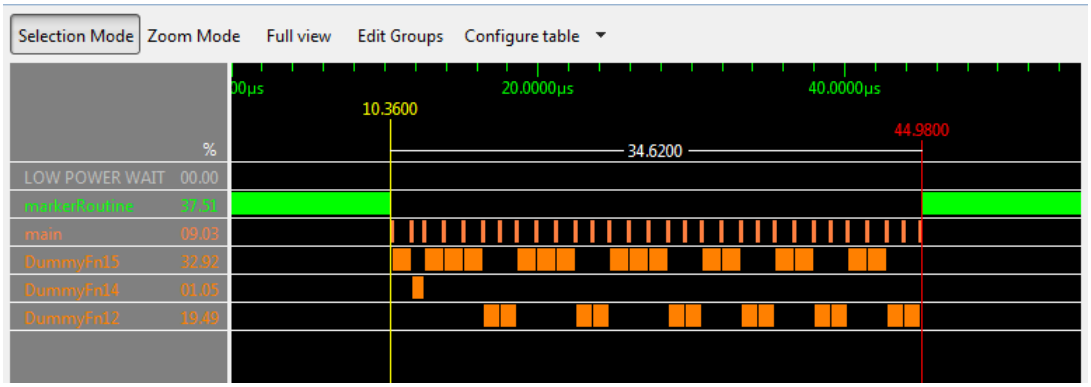

Figure 5.10: Trace of the execution time of the floating-point Kalman filter algorithm. The floating-point execution time is shown in orange sandwiched in between two functions, shown in green, placed to mark the algorithm out.

Having verified the functionality of the above algorithms, the algorithms were converted into embedded C code. The floating-point arithmetic was replaced with fixed-point arithmetic [29]. This was done because the MCF51JM128 node processor does not have a floating-point unit and therefore floating-point calculations are carried out through firmware algorithms. This increases the execution time of floating-point calculations substantially because the firmware implementation is computationally expensive. Therefore, in order to reduce the execution time of the algorithm, fixed-point arithmetic is required. Both the floating-point and fixed-point versions of the algorithm were profiled on the node hardware. Figure 5.10 shows a trace of the floating-point Kalman filter algorithm execution time. The algorithm is only functioning in one dimension, however, the execution time for all three dimensions would simply be a multiple of three of the measured execution time. In Figure 5.10 one can see the effect of the firmware routines used to implement floating-point arithmetic in the three dummy routines Fn15, Fn14 and Fn12. The total execution time of the floating-point Kalman filter algorithm is 34.62µs. Figure 5.11 shows a trace of the fixed-point Kalman filter algorithm. Again the algorithm is only functioning one dimension. The total execution time of the fixed-point Kalman filter algorithm is 14µs. This is 2.47 times faster than the floating-point algorithm. In three dimensions this equates to a fixed-point execution time of 42µs and a floating-point execution time of 103.86µs.

This testing confirms the ability of the flight computer to carry out functions F3, F9,

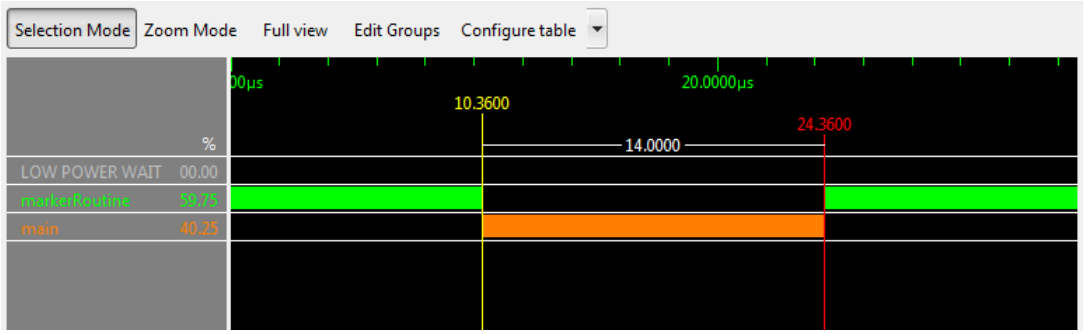

Figure 5.11: Trace of the execution time of the fixed-point Kalman filter algorithm. The fixed-point execution time is shown in orange sandwiched in between two functions, shown in green, placed to mark the algorithm out.

F10, F13 and F14.

Environmental Testing

Due to a lack of access to the correct facilities within the time frame of the project, temperature and vibration testing were not possible. It should be noted that these two tests are the most important environmental tests for rocket flight. Care needs to be taken to ensure that the temperature variations in flight do not cause large variations in the frequency of operation of the different nodes. To this end the final prototype of the flight computer should make use of temperature-compensated crystal oscillators. However, care also needs to be taken to ensure that the crystals used are able to survive the vibration environment. This is due to the fact that crystal oscillators can shatter in high vibration environments.

The only testing opportunity available within the time frame of the project was vacuum testing of the circuit boards. This was done in order to ensure that the components on the circuit boards would not out-gas when placed in a vacuum. In order to test this a circuit board was placed within a vacuum chamber, and the pressure was reduced to below 0.1kPa. The circuit board within the vacuum chamber utilised the node processor and associated electronics such as a crystal oscillator. The circuit board was made to cycle through its RAM whilst in the vacuum chamber and output this to a monitoring computer. The test was repeated three times and each time the circuit

board was monitored to ensure that it was functioning correctly. This test verified the ability of the electronics in the flight computer to operate in the atmosphere at an altitude of 50km. A full test making use of the whole flight computer in the vacuum chamber should be performed but this was not possible in the vacuum chamber that was used.

Function and Requirement Verification

The following section details how the functional requirements as well as the stresses and constraints in Requirement R13, detailed in chapter 3, have been met.

R13.1 - Size In order to meet requirement 13.1, the flight computer circuit boards have been designed with a maximum width of 52mm and a maximum length of 90mm. In Figure 5.3 the flight computer is shown fitted on a structure able to fit in a payload bay with a maximum length of 240mm.

R13.2 - Mass Prototype 3 shown in Figure 5.3, weighed approximately 450g. This included two battery packs, the master node, the recovery node, the GPS node, the telemetry node and the IMU node.

R13.3 - Power Usage The flight computer draws a maximum of 200mA per node, therefore it draws a total maximum of 1A. This equates to a total of 1667mAh required for 100 minutes of operation. Therefore the flight computer is run from two 1800mAh battery packs. A test of the operation life of the flight computer with these batteries was not conducted because not all the nodes were fully populated as the final flight computer will be. This would render a battery life test as only a partial fulfilment of the testing procedures.

R13.4 - Acceleration Due to the limitations in access to environment testing equipment mentioned above it was not possible to test the flight computer in a sustained, controlled 50g acceleration.

R13.5 - Temperature Due to the limitations in access to environment testing equipment mentioned above it was not possible to test the flight computer in a temperature chamber.

R13.6 - Electrical Isolation For the prototypes of the flight computer the use of solder mask on the circuit boards was deemed too expensive. However, to help fulfil this requirement the circuit boards should be solder masked to provide a measure of electrical isolation to the circuit boards. Further, the circuit boards should be sealed to prevent the condensation caused by rapid temperature changes from shorting pins.

R13.7 - Mechanical Robustness In order to meet Requirement 13.7, the circuit boards of the flight computer were designed with up to 4 3-mm diameter mounting holes each. This allows the flight computer to be robustly mounted as shown in Figure 5.3.

R13.8 - Noise Immunity In order to meet Requirement 13.8 the flight computer was designed using a differential communications bus. Further, the telemetry link was designed with error detection and correction in order to ensure data integrity. No tests were carried out to ensure the functionality of the flight computer in a high electromagnetic radiation environment.

F1 - Inertial Measurement Function 1 was accounted for by designing the inertial measurement unit, as per Figure 4.10, with measurement in each rotational axis with a MEMs gyroscope and measurement in each linear axis with a MEMs accelerometer. However, soldering these components onto the circuit boards was not possible. This was due to the lack of access to the equipment required to solder packages such as QFN packages.

F2 - Pressure Measurement Function 2 was accounted for by designing the inertial measurement unit to include an absolute pressure sensor. Prototype 3 included the fabrication of an inertial measurement unit circuit board. A pressure sensor was included

in this prototype, and the output of the pressure sensor was verified at sea level. Due to a lack of access to a pressure chamber the procedures for testing function 2 were not possible in the time available.

F3 - Position Measurement Function 3 was accounted for in the design of the flight computer by the inclusion of both a GPS node and an inertial measurement node. The GPS node was tested using a passive cylindrical patch antenna as one would typically find on a sounding rocket. The tests were carried out in an open area outside to simulate the conditions of a launch pad. Testing included decoding the NMEA GPS data stream and monitoring the GPS node for GPS satellite lock. The GPS node was able to lock with up to 7 satellites when outside of a building. The values of the GPS coordinates were compared to the known values of the location found through Google Earth.

In order to determine position from F1 and F2, filtering algorithms were written as detailed in section 5.2.1. These algorithms were only tested in one dimension, however, their extension to three dimensions is trivial as each axis is independent of the other two axes. The difficulty, however, in the development of these algorithms is a lack of access to test data that includes measurements for 3 linear and 3 rotational axes measurements. Due to the lack of test data it was not possible to test these algorithms as laid out in the test procedures for function 3.

F4 - Temperature Measurement Function 4 was accounted for in the design of the flight computer by the inclusion of a temperature sensor on the inertial measurement unit. A temperature sensor was included on the Prototype 3 inertial measurement unit circuit board, and its output was verified at room temperature. However, it was not possible to properly test and verify the operation of this sensor due to a lack of access to a temperature chamber in the time available.

F5 - Power Measurement Function 5 was account for in the design of the flight computer by the inclusion of current sensing on each of the power channels of the flight computer. Verification of this feature was achieved by attaching a dummy load

to each of the channels simultaneously. The output of the current sensors was measured by the ADCs of the master node, to which they were attached. The data was then forwarded through the telemetry node, in the flight computer data frame, to the ground-station and viewed through the ground-station GUI. The outputs of the current sensors were verified and compared to the expected values measured using a multimeter. However, there were some difficulties in prototype 3 with the ADC channels not correctly measuring the input voltage from the current sensors. This problem affected the measurement of voltages as well. It is marked for resolution in future work.

F6 and F21 - Operation Monitoring and Status Indication Function 6 was taken into account in the design of the flight computer firmware as shown in section 4.3. Each node was given a space on the CAN bus in which to transmit an update message as shown in Figure 4.12. If the master node does not receive a status update message from one of the nodes in the assigned period, the master node changes the status of the flight computer to show that the node is malfunctioning. Before launch, the flight computer will indicate its status through the master node buzzer. In order to test this, the flight computer was tested with nodes removed from the CAN bus. When the flight computer did not receive an update message from the disconnected node, it changed the buzzer beep to indicate a faulty node, thus verifying both function 6 and function 21.

F7 and F8 - Data Storage and Transmission Function 7 was accounted for in the design of the flight computer through the inclusion of an SD card on the master node of the flight computer. In order verify the operation of the SD, card a known dataset using a repeating pattern was written to the SD card. The contents of the SD card were then read out by the master node and compared to the data written to the SD card. The datasets were found to be the same thus verifying the operation of the SD card.

Function 8 was accounted for in the design of the flight computer through the inclusion of a telemetry node. The operation of the telemetry link with the ground-station was verified by setting up a telemetry link between the ground station and the flight computer and then monitoring the output of the flight computer through the ground-

station GUI. The data output was found to be the same as when connecting to the flight computer directly via a USB cable.

In the firmware each data frame is both placed into a buffer on the master node, for storage on the SD card, and transmitted from the telemetry node to the ground. The operation of this was tested by running the flight computer on the bench, and monitoring both the output of the flight computer on the ground-station GUI, and the writing of data to the SD card.

R17 and F9 to F14 - Event Detection The testing and verification of these functions is detailed in section 5.2.

F15 - Dangerous Trajectory Detection This function was not implemented due to time constraints on the project. In order to implement this function, the flight computer needs to keep track of its horizontal displacement, that is, its displacement perpendicular to its altitude. A maximum horizontal displacement would be set before launch on the flight computer, and the current displacement would be measured against the displacement limit.

F16 to F18 - Event Initiation Functions 16 to 18 were accounted for in the design of the flight computer by the inclusion of a recovery node. In order verify the functionality of each of the igniter channels on the recovery node, each channel was attached to a resistive load with an LED. The channels were cycled on and off and the LEDs were monitored to ensure the switching worked as intended. Further, the charge pump on the recovery node was tested by enabling it and testing the voltage output across the terminals of each of the igniter channels. The triggering of the events in functions 17 and 18 were not simulation tested due to a lack of multi-stage rocket flight data. The ability of the recovery node to provide sufficient current was not tested due to the time constraints on this project.

F20 - In-Flight Commands Function 20 was implemented through the telemetry and USB links to the ground-station computer. Through these links the user is able

to control and configure the flight computer both on the ground and in-flight. This functionality was tested by sending a command to the flight computer requesting that individual power channels be switched either on or off. Each power channel was then monitored using a resistive load and an LED to indicate whether it was on or off. The power channel switching responded correctly when commands were sent to the flight computer. This verified the operation of the in-flight commands as per the testing procedures.

5.4 Failure Modes, Effects and Preventative Measures

The following section details potential failure modes and their effect on the operation of the flight computer.

Failed Power Supply The first and most critical failure is a failure of the main power supply system. A failure in the main power supply would mean a loss of power for the entire flight computer system. This failure would disable the flight computer completely and thus cause mission failure. To mitigate the chance of this failure, a second battery back has been included in the design of the flight computer. In the event of a battery failure on the main battery pack, the flight computer will seamlessly switch over to the reserve battery pack.

In order to prevent mission failure in the event of this failure, critical systems are provided with backup power supplies. The correct operation of the recovery node is essential for the deployment of parachutes and thus the recovery of the rocket. In order to prevent a main power supply failure from disabling the recovery system, the recovery node has been provided with a backup power supply independent of the main power supply.

The current measures to ensure the operation of the recovery node may be improved with the following recommendations. In the event of such a failure, the data required to determine apogee would no longer be available from the inertial measurement unit. Therefore a backup countdown to apogee based on the pre-flight simulation of the

rocket flight should be implemented. Further, to increase redundancy, a single Z-axis accelerometer should be included on the recovery node to provide a rough estimation of apogee.

Failed Communication Bus Another potential failure is a break in the CAN bus. The CAN bus is used as the main communication bus of the flight computer. In the event of a broken wire or pin header a node or nodes could be cut off from the rest of the flight computer. In the worst case, the recovery node would be cut off from the inertial measurement unit's determination of apogee. This failure would then lead to mission failure as the recovery system would be unable to deploy the parachutes at the correct time.

There are currently no measures in place to stop this failure from causing a mission failure. In order to do so, the recovery node should be equipped with independent ways to calculate apogee. This may be through a time based system as mention above, or through the addition of simple inertial measurement as mentioned above. A further measure to reduce the risk of a complete communication breakdown would be to include a redundant CAN bus. In this case a failure on one bus would not cause any nodes to be cut off from the rest of the flight computer because communication would still flow through the second bus.

Babbling Idiot Node A babbling idiot failure is where a single node continually transmits corrupted data onto the communication bus. The malfunctioning node floods the communication bus with corrupted data potentially preventing the other nodes from transmitting on the bus, and severely reducing the available bandwidth on the bus. The effect of this failure would be similar to that of a failed communication bus. The individual nodes of the flight computer could be cut off from one another.

In order to reduce the chance of a complete mission failure, the recovery node should be provided with independent ways of calculating apogee as mentioned above. Further, one method of preventing a babbling idiot node from disrupting communication on the bus is to separate the node from the communication bus with a bus guardian [30]. One implementation of such a system only allows a node to transmit on the bus

when it is scheduled to. This prevents the node from flooding the bus and thus prevents this failure mode.

Loss of Telemetry A further potential failure is the loss of telemetry. This may be through the failure of the telemetry node or ground-station, or from a loss of connection between the ground station and the telemetry node. This failure only has the potential to cause mission failure in the event that apogee is not detected by the flight computer. This is because it would prevent a manually induced deployment of parachutes.

In order to prevent this failure from causing a mission failure, the recovery node should be provided with a means of detecting apogee that is independent of inertial measurement. One method is detailed above and involves the use of a count down to a pre-calculated time-to-apogee. The calculation is done through means of a pre-flight simulation.

Node Failure The last potential failures to consider are the individual node failures. The recovery node and telemetry/ground-station nodes are excluded because their failure has already been considered above.

- **Master Node Failure:** The failure of the master node would induce a system wide power supply failure. This would lead to complete mission failure as detailed above in the discussion of a failed power supply. A further consequence of this failure is the loss of timing messages on the CAN bus. This would cause any nodes still operating to drift apart in timing. In the event of more than one node with backup power, a second redundant timing system could be put in place through one of the still operating nodes. In the event of not receiving triggering messages after a pre-defined time, this node would then begin to transmit the timing messages.
- **GPS Node Failure:** The failure of the GPS node has no major effect on the operation of the flight computer. In future work the GPS coordinates are intended

to be fused with the inertial measurement data to improve the position approximation. Loss of this data would thus cause the approximation of position to deteriorate.

- **IMU Node Failure:** The failure of the IMU node would mean the loss of the ability of the flight computer to detect in-flight events. This failure will lead to mission failure in the event that the flight computer cannot detect apogee as discussed above. Further potential event-detection related failures would include the failure to detect motor-stage burnout and the subsequent failure to initiate new motor-stage ignition. This would prevent the rocket from reaching the intended altitude. In order to prevent these failures causing mission failure, independent inertial measurement should be included on the recovery node to provide a backup means of detecting these events.

5.5 Safety and Reliability

The following section deals with the safety of the flight computer when human interface is required and the potential measures to be taken for safe and reliable operation of the flight computer.

Launch Safety

If the flight computer is used as the device to initiate launch, then the most dangerous human interface with the flight computer will be after the connection of the motor to the rocket. In the event of an accidental ignition of the motor, the user may be badly burned or even killed depending on the size of the motor. In order to prevent this, a number of safety locks have been included on the recovery node. The recovery node is the node that is responsible for the firing of igniters to trigger launch or other events. As mentioned in the recovery node feature list of chapter 4, the recovery node has a set of interlocks that are designed to prevent the accidental ignition of a rocket motor. The process required to arm and fire a motor stage is as follows:

1. Arm enable logic signal high.
2. Next the fire enable logic signal high.
3. Next the user needs to select an igniter channel and enable it with a logic signal high.
4. Finally the user needs to issue the fire command via a logic high on the fire line. Only with all the other interlocks at a logic high will this command work.

Each logic signal high switches a solid state switch on the recovery node.

A further safety measure is the inclusion of an extension to the recovery node that allows the user to enable or disable each igniter channel individually with a hardware jumper. The hardware jumper provides a physical disconnection of the current flow to the igniter. This extension also includes a visual indication of which logic signals are high. This feature is to warn a user of the danger of a live igniter channel.

A shortcoming of this design is the fact that the user must physically close the jumper, to enable the firing of the motor, whilst on the launch pad. This still presents a safety hazard for the user. One potential solution would be to automate the closing of the igniter jumper with a small servo motor controlled via an independent wireless link. A second solution would be to change the configuration of the jumper so that removal of the jumper enables the igniter channel. In this case, a physical disconnection between the igniter and the current source could be made with a normally open relay. This would enable the user to move a safe distance away from the launch site, and then pull a lead which then enables the launch system.

Igniter Ignition Safety

A second human interface with the rocket, that presents a safety hazard for the user, is the connection of the igniters themselves. The accidental activation of an igniter may cause harm to the user through electrical shock or burns. The same measures discussed for launch safety apply to the safe connection of the igniters themselves.

The Use of Reliable C Compilers

Normally in the aerospace industry to ensure the safe and reliable operation of firmware, programming standards and certified compilers are used. This is to prevent the programmer from using any features of the language that may be ambiguous or lead to undefined behaviour, and to prevent the programmer from using any features of the language that have the potential to fail. One such standard is the MISRA C standard [31]. Due to the cost of using such a standard, and an associated compiler, no standards or certified compilers have been used in the development of this flight computer.

Error Correction Via Voting

Often in aerospace systems, error detection and correction are achieved through voting. This is normally done by triplicating the system so that three independent outputs of the same system are available. These outputs are then connected to a voting system. The voting system compares the outputs of each of the systems and allows the majority output to propagate. In this way errors are prevented and the overall reliability of the system is increased. In the case of the SAAO flight computer, the space, weight, power and budget constraints on the project have precluded this as a general reliability solution. However, in future revisions of the flight computer, inclusion of this methodology should be considered for the increased reliability of the recovery node as it is the most critical system to the safe and reliable use of a sounding rocket.

5.6 Conclusion

In this chapter we have presented the implementation of the flight computer in hardware prototypes, firmware and the ground-station GUI. We then described the testing carried out on these prototypes to verify the functions detailed in chapter 3. Finally, examined the potential failure modes of the flight computer and their effect on the operation of the flight computer. We have detailed the measures taken to prevent these failures or we have suggested measures to be taken in future work. Finally, we have

looked at potential safety and reliability issues in the use of the flight computer and, where possible, we have suggested solutions to these issues.

Chapter 6

Conclusions and Future Work

6.1 Introduction

The following chapter gives an overview of the achievements and results of this dissertation. We also note the features that were not considered for this project due to time or budget constraints. Based on these features and the achievements of this dissertation we then make suggestions for future work in the project.

6.2 Achievements

In this dissertation we have developed a flight computer for sounding rockets. We started by reviewing two projects of similar intent in order to ascertain how other people have approached this design problem. We then analysed the requirements and constraints, for the flight computer, provided by SAAO. From those requirements we derived the functions that the flight computer is required to carry out. We then presented the testing procedures required to verify that the flight computer is able to carry out the required functions and operate within the constraints provided by SAAO. With this background we then designed the flight computer, starting with the overall architecture of the flight computer, and then focusing on each of the flight computer

nodes. This design included the design of the hardware of the nodes as well as the design of the operating system and algorithms to run on the flight computer hardware. The hardware design consisted of two parts, first the digital hardware design for the nodes, and second, the RF simulation and hardware design of the RF output staqe. Once the design was completed we began developing a series of prototypes, each one a step forward in achieving the functionality required of the flight computer. Once we had produced a working prototype, we began the development of the time-triggered operating system of the flight computer, as well as the ground-station graphical user interface. We also developed event detection and filtering algorithms which we simulated and tested on a desktop computer before optimising them and running them on the flight computer itself.

Due to the time and resource constraints on the project a few features were not implemented in this dissertation. These include:

- Algorithms for the detection of separation and new-stage-ignition, as per F11 and F12, were not developed due to a lack of multi-stage rocket flight data. It would not have been possible to test the detection of these events, thus it was decided to omit them from the firmware for the moment.
- An algorithm for detection of a trajectory outside of nominal trajectory, as per F15, was not developed. This was due to time constraints on the project. Such an algorithm requires the implementation of an attitude filter in order to determine accurate horizontal displacement.
- F17 and F18, the separation of rocket stages and the ignition of rocket stage motors, were not simulation tested due to a lack of multi-stage rocket flight data. The ability of the firing mechanisms to supply sufficient current to fire the igniters was not tested either. This was due to the time constraints on the project. However, the mechanisms to trigger the firing of the igniters to enable initiation of these events were tested and verified.

6.3 Future Work

A number of features laid out in the requirements of SAAO were not completed due to the time constraints on this dissertation. These features were relegated to future work and are documented here.

Features

The following features should be added to the flight computer in future work.

Flight Termination In order to launch rockets above a certain altitude at a test range, the rockets require an independent flight termination module. A flight terminate module would allow the user to terminate the flight of the rocket. This is to ensure that a rocket that goes off the predicted flight path will not cause damage to any persons or property. This should be done in line with the requirements of SAAO namely requirement R11.

GPS A COTs GPS receiver was used in this project. However, in order for the GPS to function under all possible flight conditions, a GPS receiver not constrained by the ITAR regulations is required.

Ground Station GUI The ground-station graphical user interface needs to be extended to provide clear visual information to the user. The ability to simulate and plot the rocket's trajectory should be included as well as the ability to plot the actual trajectory of the rocket in real time over the simulated trajectory. This will provide the user with clear information with which to make decisions with regard to the flight termination module.

Complete Inertial Measurement and Guidance and Control Due to time constraints and the lack of access to the facilities required to manufacture properly the

inertial measurement unit, a full 6-degree of freedom inertial measurement unit was not constructed. This will need to be addressed in future work. Once manufactured, the testing procedures outlined in section 3.6 for F1 should be carried out to verify the correct operation of the inertial measurement unit and its algorithms.

In order to improve the approximation of position done through the inertial measurement unit's filtering algorithms, the GPS coordinates measured on the GPS node should be fused with the position approximation data obtained through the inertial sensors.

Once manufactured and tested, the flight computer's inertial measurement unit will provide the platform from which a rocket's trajectory may be controlled, for example, via gimballing a rocket motor. Any mission requiring the precision placement of a payload at a specific location would require this functionality. Proof of the inertial measurement unit's ability to fulfil this requirement could be accomplished through using the inertial measurement unit to actuate control surfaces.

Requirements and Constraints

Electrical Isolation The current prototypes of the flight computer are not solder-masked or sealed with a conformal coating. In order to improve the electrical isolation of the flight computer circuit boards, the circuit boards should be solder-masked. This will provide isolation of the circuit board tracks. Further isolation and protection may be provided by sealing the circuit board with a conformal coating. This coating will also protect the circuit board against shorts from the condensation that will occur during the rapid temperature changes the flight computer will undergo.

Size In order to reduce the size of the flight computer, which will allow it to placed in even smaller diameter rockets, the circuit boards should be redesign with four layers. This will provide a significant reduction in the width and length requirements of the circuit boards.

Functions

Detection of Stage-Separation and New Stage-Ignition Algorithms for the detection of stage-separation and new stage-ignition should be developed as per F11 and F12. In order to test and refine these algorithms, multi-stage rocket flight data will be required.

Triggering of Stage-Separation and New Stage-Ignition In order to fulfil the requirements of F17, the detection of motor burnout should be set up to trigger stage-separation for rockets that use pyrotechnics to separate stages.

The detection of stage-separation should be set up to trigger the firing of an igniter for new stage-ignition. This is in order to fulfil F18. Finally, the ability of the firing mechanism to provide the current necessary to fire the igniters should be tested to verify fully functions F16 to F18.

Detection of Outside Nominal Trajectory In order to fulfil the requirements of F15, an algorithm should be written to detect trajectories that are outside a given safe zone. As a pre-requisite to this algorithm, an attitude filter should be written in order to accurately determine the attitude of the rocket. Further, the constant gain Kalman filter used to determine the position of the rocket should be extended to three dimensions, as mentioned above. Through these additions and extensions the flight computer would be able to determine the horizontal displacement of the rocket. This would enable the detection of a displacement that would place the rocket outside a given zone of safe operation.

Environmental Testing

Due to a lack of time and access to facilities the environment testing laid out in section 3.6 was not done. This testing is critical to verifying the ability of the flight computer to carry out F1 to F21 reliably. In particular testing of the flight computer under the temperature and vibration environment expected in rocket flight is essential as both

of these variables have the potential to impair severely the functioning of the flight computer.

Crystal Oscillators In order to ensure that the flight computer operates reliably with the temperature variations experienced during rocket flight, the crystal oscillators currently used should be replaced with temperature-compensated crystal oscillators. This will ensure that the in-flight temperature variations do not cause a large drift in the operational frequency of each of the flight computer's nodes. This is important because a variation in the frequency of operation of the nodes will alter the sample time of the sensors and thus the algorithms for the calculation of acceleration, velocity and position will begin to produce erroneous outputs.

Failure Modes, Reliability and Safety

Section 5.4 and 5.5 presented a number of failure modes, reliability issues and safety issues. In each case, a solution was suggested either to prevent a failure mode or to improve reliability and safety. In future work the suggestion made in this section should be considered for implementation.

6.4 Conclusion

In this chapter we have reviewed the achievements presented in this dissertation. We have also discussed the features that did not see implementation or were only partially implemented or partially tested in this dissertation due to time or other resource constraints. Based on both of these, recommendations were then made for future work to improve the flight computer.

Appendix A

RF Measurements

The following were the conditions of testing and the S parameters measured by the network analyser.

Power Amplifier The power amplifier was tested with the following setup:

- Input Power: 0dBm
- Supply Voltage: 3.3V
- Calibration with 25dBm attenuator

Low Noise Amplifier The low noise amplifier was tested with the following setup:

- Input Power: -30dBm
- Supply Voltage: 5V
- Calibration with 25dBm attenuator

SPDT The SPDT was tested with the following setup:

- Input Power: 0dBm
- Switching Voltage: 1.8V
- Calibration with no attenuator

The following figures show the measured S11, S21 and Isolation of the individual low noise amplifier, power amplifier and SPDT components.

Power Amplifier S11

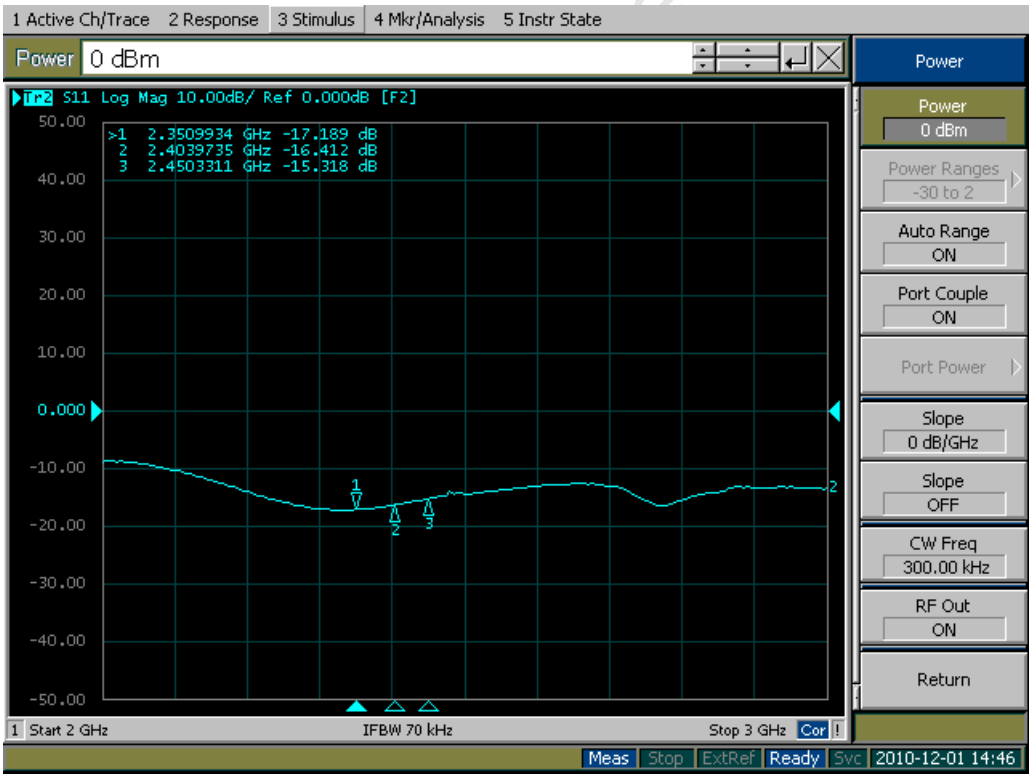

Figure A.1: Measured power amplifier S11.

Power Amplifier S21

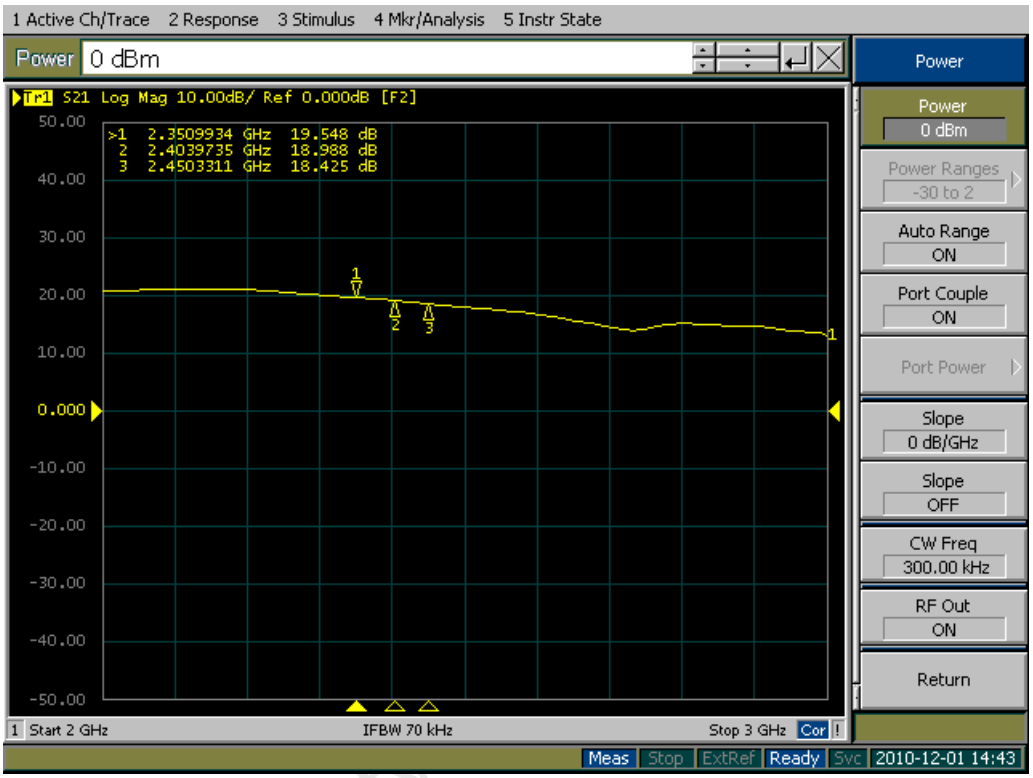

Figure A.2: Measured power amplifier S21.

Low Noise Amplifier S11

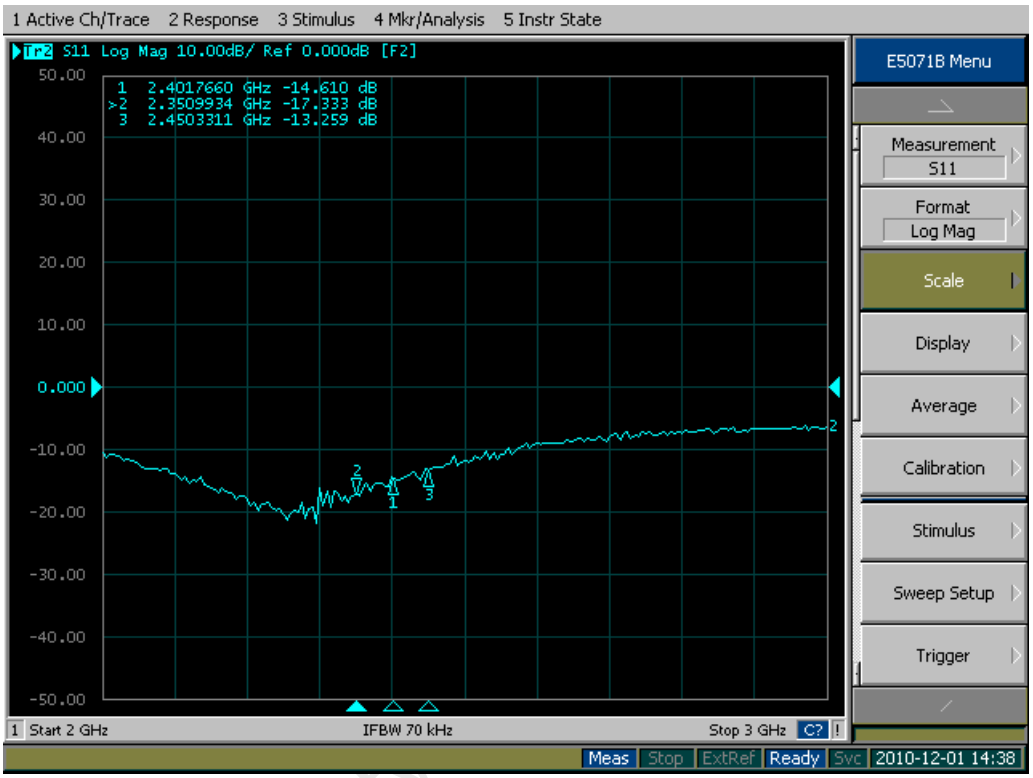

Figure A.3: Measured low noise amplifier S11.

Low Noise Amplifier S21

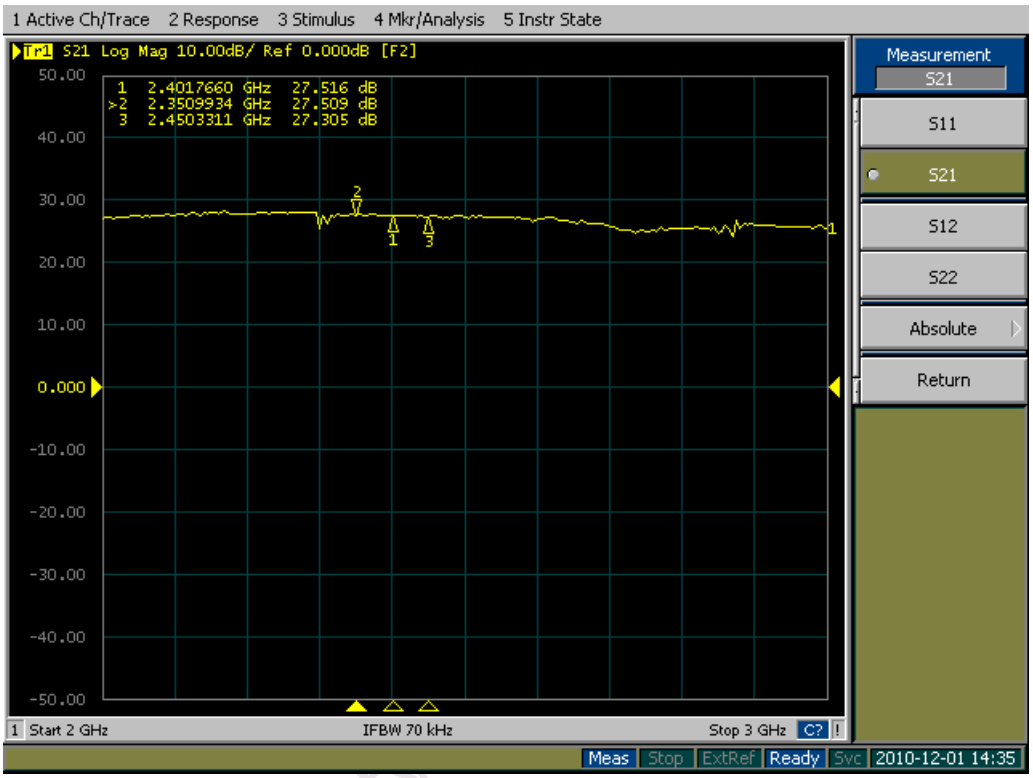

Figure A.4: Measured low noise amplifier S21.

Single Pole Double Throw Switch S11

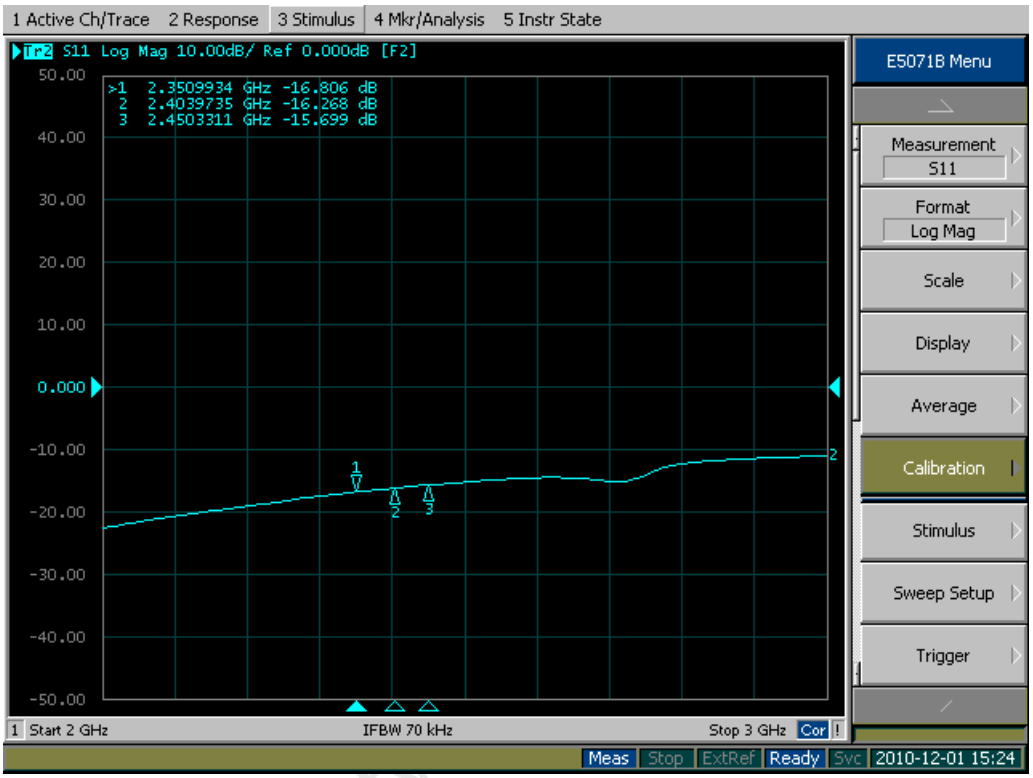

Figure A.5: Measured SPDT S11.

Single Pole Double Throw Switch S21

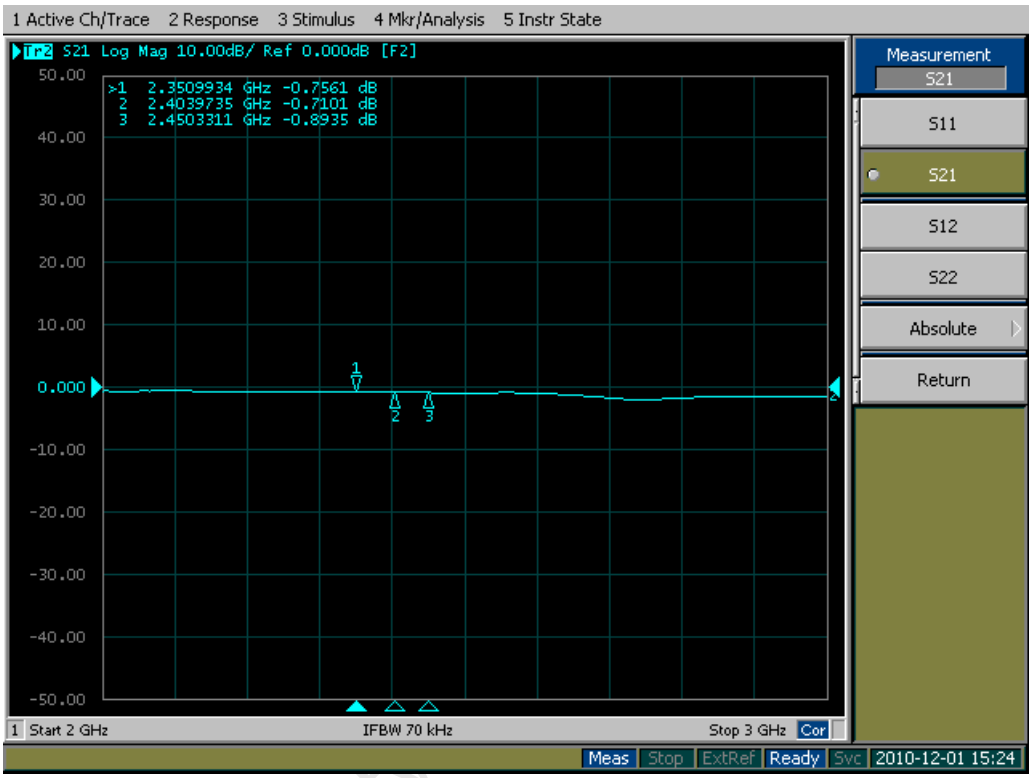

Figure A.6: Measured SPDT S21.

Single Pole Double Throw Switch Isolation

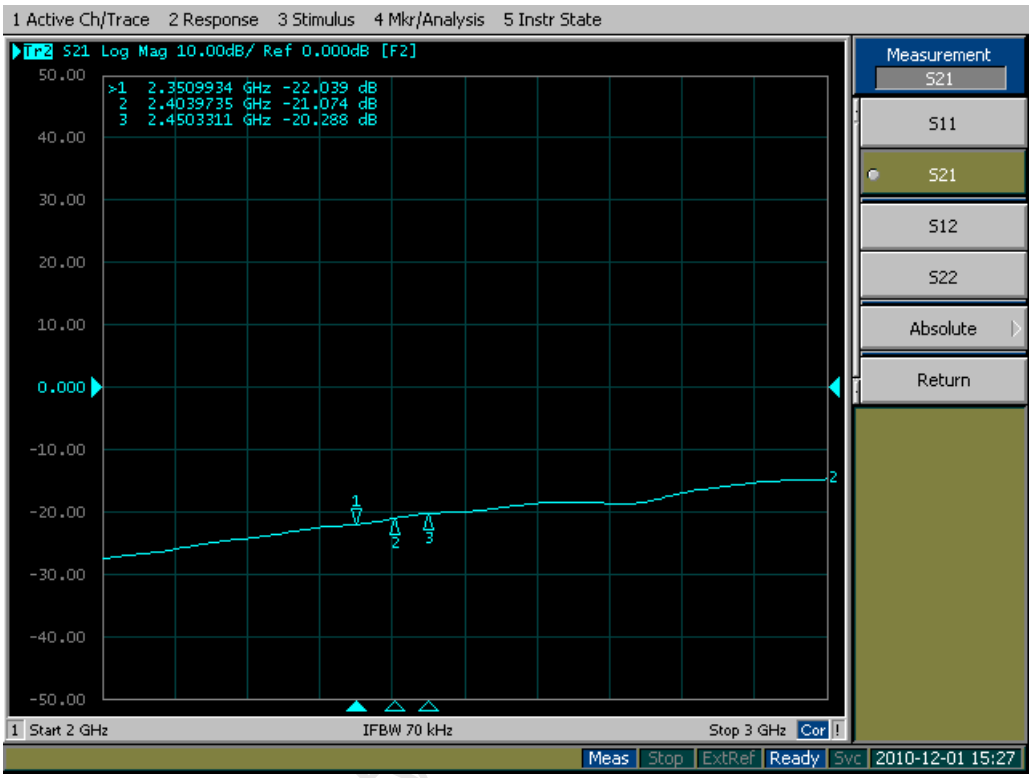

Figure A.7: Measured SPDT isolation.

Appendix B

CAD Files and Code

The schematics and board designs for the both the RF hardware and the digital hardware may be found on the attached CD. Also included on the CD are the firmware drivers for each of the nodes as well as the operating system files for the flight computer as a whole. The graphical user interface code is also included with the files required for it to compile and run. The following directory structure may be found on the CD:

- Appendix B Folders
  - CAD Files
    - Digital CAD Files
    - RF CAD Files
  - Firmware
    - Peripheral Drivers
    - Operating System
  - GUI Work
    - Resources
    - Test_UI
    - Test_UI-build-desktop

Bibliography

[1] Elaine M. Marconi. *What is a Sounding Rocket?* URL: http://www.nasmailv/missions/research/f_sounding.html.

[2] *NASA Sounding Rocket Program Overview*. URL: http://rscience.gsfc.nasa.gov/srrov.html.

[3] *GWIZ Flight Computers Website*. URL: http://www.gwiz-partners.com/index.html.

[4] *RDAS Flight Computers Website*. URL: http://www.aedelectronics.nl/rdas/index.htm.

[5] *Portland State Aerospace Society Website*. URL: http://psas.pdx.edu.

[6] AED Electronics. *RDAS User Manuel*. V1.5. AED Electronics. Bertelindislaan 3, 5581 CS Waalre The Netherlands, 2001.

[7] S. Bailey, J. Davidson, and G. LeBrasseur. *PSAS Avionics Node Front End for LV2b Rocket*. Tech. rep. Portland State Aerospace Society, 2006.

[8] *A Spiral Model of Software Development and Enhancement*. IEEE, 1988.

[9] University of Leicester Leicester U. K. Michael J. Pont. *Patterns for time-triggered embedded systems: building reliable applications with the 8051 family of microcontrollers*. 1st. ACM Press/Addison-Wesley Publishing Co. New York, NY, USA, 2001.

[10] *Fault-Tolerant Time-Triggered Communication Using CAN*. IEEE, 2007.

[11] Philips Semiconductors. *The I2C-Bus Specification*. 2000.

[12] *eCos Kernel Overview*. URL: http://ecos.sourceware.org/docs-1.3.1/ref/ecos-ref.4.html.

[13] Mike Jones. *What Really Happened On Mars*. URL: http://research.microsoft.com/en-us/um/people/mbj/mars_pathfinder/mars_pathfinder.html.

[14] H. Himelblau et al. "Dynamic Environmental Criteria". In: vol. 7005. NASA Technical Handbook. 2001. Chap. 9.

[15] United States of America Department of Defence. *Test Requirements for Launch, Upper-Stage, and Space Vehicles*. Military Standard. 1994.

[16] Electronic Industries Association. *Standard for Electrical Characteristics of Generators and Receivers for use in Balanced Digital Multipoint Systems. EIA Standard; RS-485*. 1983.

[17] Robert Bosch. *CAN Specification 2.0*. 1991.

[18] Freescale Semiconductors. *MCF51JM128 ColdFire Integrated Microcontroller Reference Manual*. Revision 2. 2009.

[19] M. Milandri. *Development of a Flight Computer for Sounding Rocket Applications*. BEng Thesis. The University of Cape Town, 2009.

[20] Nordic Semiconductors. *Reference Design, External PA for nRF24xx*. Revision 1.0. 2005.

[21] Nordic Semiconductors. *nRF24L01+ Product Specification*. Revision 1.0. 2008.

[22] Avago Technologies. *MGA-83563 Data Sheet*. 2011.

[23] Avago Technologies. *MGA-86563 Data Sheet*. 2010.

[24] CEL. *GaAs Integrated Circuit UPG2214TB*. 3rd Edition. 2004.

[25] Portland State Aerospace Society. *A Quick Derivation relating altitude to air pressure*. Version 1.03. 2004.

[26] David Schultz. *Application of the Kalman Filter to Rocket Apogee Detection*. Research and Development Report. NAR 63255. National Association of Rocketry, 2002.

[27] Spectrum Elektrotechnik GmbH. *SMA Specifications to MIL-C-39012*.

[28] Josef Wilgen Uwe Rathmann. *Qwt User's Guide 6.0.1*. URL: http://qwt.sourceforge.net.

[29] Joe Lemieux. *Fixed-point math in C*. 2003. URL: http://www.eetimes.com/discussion/other/4024639/Fixed-point-math-in-C.

[30] Giuseppe Buja, Juan R. Pimentel, and Alberto Zuccollo. "Overcoming Babbling-Idiot Failures in CAN Networks". In: *IEEE Transactions on Industrial Informatics* 3.3 (2007), pp. 225–231.

[31] *The Motor Industry Software Reliability Association*. URL: http://www.misra.org.uk.

[32] David Schultz. *Barometric Apogee Detection Using the Kalman Filter*. Research and Development Report. NAR 63255. National Association of Rocketry, 2002.

[33] Michael J. Pont Michael Short Mouaaz Nahas. "Reducing message-length variations in resource-constrained embedded systems implemented using the Controller Area Network (CAN) protocol". In: *Journal of Systems Architecture* 55 (2009), pp. 344–354.

[34] Jim Zyren and Al Petrick. *Tutorial on Basic Link Budget Analysis*. 1998.

[35] Atmel Corporation. *Integration of an External Low-noise Amplifier, Improving the Sensitivity of the Receiver*. 2006.